# WAVELET-BASED AUDIO EMBEDDING & AUDIO/VIDEO COMPRESSION

THESIS

Michael J. Mendenhall, Captain, USAF

AFIT/GE/ENG/01M-18

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GE/ENG/01M-18

# Wavelet-Based Audio Embedding & Audio/Video Compression

THESIS

Presented to the Faculty
Department of Computer and Electrical Engineering
School of Engineering
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Engineering

Michael James Mendenhall, B.S. Computer Engineering
Captain, USAF

March, 2001

AFIT/GE/ENG/01M-18

# Wavelet-Based Audio Embedding & Audio/Video Compression

Michael James Mendenhall, B.S.

Captain, USAF

Approved:

5 MAR 01

Maj Roger L. Claypoole Jr.
Thesis Advisor
Date

Gary Lamont
26 FEB '01

Dr. Gary Lamont
Committee Member
Date

Mark E. Oxley
5 Mar 01

Dr. Mark E. Oxley
Committee Member
Date

## *Acknowledgements*

I would like to take the opportunity to thank friends and colleagues I have had the privilege to work with while here at AFIT.

I would like to thank Major Claypoole, Professor Gary Lamont, and Dr. Mark Oxley for partaking in my thesis and making sure I was doing all the right things. Special thanks to Major Claypoole for his insight, knowledge base, and help. By the way, EENG 680 was without a doubt the best class I have ever taken.

Thanks to Professor Lamont for his advise with this thesis document. Your experience was very valuable to me and I greatly appreciated your input. And to a lessor extent, I am glad I was able to be one of your Advanced Algorithm students. Although it was far more work than any person should have to do in one quarter, I learned a lot of valuable skills I am sure will be used throughout my Air Force career.

I would like to thanks to Rob Bonner. He helped me obtain and format the audio/video data used for my thesis. Without his help, I would have had a real hard time getting an audio/video sequence to test my audio embedding audio/video compression system on. Rob, you have been a great study partner, lab partner, and friend over the past year and a half. and I will be sure to keep in touch with you and .

Thanks to our friends . I met the family in January of ‘97 in the booming city of Biloxi Mississippi while making my way through all of the challenges BCOT had to offer. BCOT successfully turns useful human beings (engineers and such) into the living dead. Aside from the lack of mental challenges BCOT had to offer, I had fun goofing off with While here at AFIT, I had the opportunity to take a couple of classes with Most notably was Advanced Algorithms. Although painful, I had a great time working with you on those thesis length projects Dr. Lamont liked to assign. And I can say with confidence, we compliment each other quite well and are a terrific team. Hopefully someday we will have the opportunity to work together again. Maybe next time we get together we can play some golf.

I would like to express my appreciation for my friendship with . We met in the Fourier short course during the summer of ’99 and have had the (miss)fortune of many classes together. : and I have become very close friends. I appreciate the time spent in the gym lifting, running, talking about frustrations with classes, thesis, and instructors. Our in-depth conversations about thesis hurdles are what really made the thesis process a success. I think we did a great job of driving each other to get things done early and done well. As a result, I think we have each developed really outstanding products. Sorry you have to stay here and were unable to get the AFPC ogre, , to change your assignment to the 84th RADES at Hill AFB, UT. Hopefully we will have the opportunity to work with each other once again. I am sure we will be seeing you down the road!

Finally, I would like to thank my wife, putting up with the strains and pains of graduate school. Its over (woo hoo), now we can get on with those things that are really important; spending time together in the evenings and playing with the dogs!!

Michael James Mendenhall

## *Table of Contents*

## *List of Figures*

Figure Page

Figure Page

## *List of Tables*

AFIT/GE/ENG/01M-18

## *Abstract*

With the decline in military spending, the United States relies heavily on state side support. Communications has never been more important. High-quality audio and video capabilities are a must.

Watermarking, traditionally used for copyright protection, is used in a new and exciting way. An efficient wavelet-based watermarking technique embeds audio information into a video signal. Several highly effective compression techniques are applied to compress the resulting audio/video signal in an embedded fashion. This wavelet-based compression algorithm incorporates bit plane coding, index coding, and Huffman coding.

To demonstrate the potential of this audio embedding audio/video compression system, an audio signal is embedded into a video signal and the combined signal is compressed. Results show that overall compression rates of 15:1 can be achieved. The video signal is reconstructed with a median PSNR of nearly 33dB. Finally, the audio signal is extracted without error.

Wavelet-Based Audio Embedding & Audio/Video Compression

# *I. Introduction*

## *1.1 Background*

Reliable real-time audio/video communication is necessary for today's military. Funding and personnel numbers have declined dramatically since the early 1980's. Many military functions of today are humanitarian and peace keeping operations. Due to this lack of personnel and decline of military funding, the military relies heavily on state side support and state-of-the-art weapon systems. Communications has never been more important. High-quality audio/video data for inter- and intra-theater battle management as well as data analysis is a must.

The amount of information in digital audio/video data is enormous. It is expensive to store and requires a significant amount of bandwidth to transmit. Many communications systems are old and do not have the bandwidth necessary to transmit high-quality audio/video data. Furthermore, audio and video signals are treated independently which causes synchronization problems at the receiving end. How are these problems solved? Bigger and better communications systems would surely help. However, these systems are expensive, take a long time to incorporate, and require maintenance. Instead of replacing existing communication systems, it easier and cheaper to process the audio and video data in a smart way.

*1.1.1 Discrete Wavelet Transform.* The discrete wavelet transform (DWT) is a popular transform used to achieve high compression rates while maintaining high image quality. The DWT performs a dyadic decomposition of a two-dimensional signal. It represents the original signal with a coarse approximation and a series of details. The DWT has several desirable properties:

1. Primary properties (15):

    (a) Locality – Wavelet coefficients are localized in time and frequency simultaneously.

    (b) Parsimony – Wavelet coefficients in images tend to be sparse.

    (c) Multiresolution – Wavelet basis functions at different scales are integer shifts and dilations of a single mother wavelet.

2. Secondary properties (14, 13):

    (a) Clustering – Wavelet coefficients adjacent to a large or small wavelet coefficient tend to be large or small.

    (b) Persistence – Wavelet coefficients with large or small values propagate across scales.

These properties make the DWT ideal for image compression and many other image processing functions. The parsimony property of the DWT is the key property for this audio embedding audio/video compression system.

*1.1.2 Digital Watermarking.* Watermarking was used as early as the 13th century to identify their maker and to prove authenticity of paper money (6). Today, these physical watermarks are used for the same purpose. The role of digital watermarking is very similar to its physical counterpart. Digital watermarking is almost exclusively used to prove ownership of digital media (i.e., copyright protection.) Digital watermarking is a process for which a secondary signal is embedded into a primary signal. It is used in this research to embed an audio signal (secondary signal) into a video signal (primary signal.)

### *1.2 Problem Statement*

A method to embed audio data into a video sequence and compress the resulting combined audio/video signal is proposed. By combining the audio and video data into a single signal, synchronization problems are alleviated because interleaving independent signals is no longer required. The size of the resulting audio/video stream is smaller for two reasons. First, the audio information is embedded in the video signal and therefore does not require storage space outside of the video data. Second, the combined audio video sequence is compressed thus the space savings are even greater.

This thesis effort investigates the use of digital watermarks to embed digital audio data into a digital video sequence. Watermark usage is in its infancy and is almost exclusively used for copyright protection. However, digital watermarks show promise for many information-hiding applications. Recently, wavelet transform based watermarking methods have proven to be a viable method of digital watermarking. Furthermore, wavelet transform based compression techniques have already proven to have superior performance over compression techniques in other domains (e.g. Fourier, DCT, etc.) Combining the two to create an audio embedding and audio/video compression system has great potential. If the audio information is embedded smartly, the integrity of the embedded audio information is maintained after video frame compression is applied.

*1.2.1 Resources.* The following resources are needed to simulate the audio embedding audio/video compression system developed during this thesis effort.

1. Hardware resources:
   (a) Processor – A Pentium III 600 MHz processor based machine is used.
   (b) Memory – More than 256 MB of RAM is required if running Matlab under Windows98. If running Matlab under Linux, 256 MB of RAM with a 256 MB swap space will suffice.
   (c) Disk space – The Red Hat 6.2 Linux install, Matlab Version 5.3.0.14912a (R11) student edition install, and space for scripts, input data, and results require approximately 2 GB of hard disk space.
2. Software resources:
   (a) The Linux operating system is recommended for machines with 256 MB of RAM or less. Memory management under Windows98 does not suffice for Matlab scripts where large arrays are operated upon.

(b) Matlab software is required. All scripts can be executed using Matlab Version 5.3.0.14912a (R11) student edition.

*1.2.2 Matlab.* Matlab is used to simulate the audio embedding audio/video compression system developed as a result of this thesis effort. Matlab is designed as a simulation environment. It has predeveloped packages ideal for manipulating matrices, performing complex mathematical operations (such as convolutions), and visualizing several forms of data.

Although Matlab is an excellent resource for developing simulations and prototyping systems, it is slow. The performance of Matlab is poor when compared to applications developed in C or C++. Matlab is an interpretive language. Poor performance is indicative of interpretive languages. However, the functionality of Matlab and the ability to easily visualize data far out way the lack of performance. For these reasons, Matlab is used to develop the simulations for this thesis effort.

### *1.3 Scope*

This research demonstrates the concept of audio embedding and audio/video compression. The overall goal of this research is to demonstrate that audio can be embedded into video and extracted after compression has been applied to the resulting audio/video signal. The focus is on overall quality of the modified video frames and the accurate extraction of the embedded audio information.

The audio data format considered in this research is a stereo format where each audio sample is quantized to 8-bits. The video data format considered in this research is 256 × 256 pixel video frames where each pixel is formatted to 8-bit grayscale. The video data frame rate considered is 30 frames/second. Color video data is not investigated.

### *1.4 Organization*

This thesis document is segmented into five chapters. Chapter 1 provides the background to the problem, the problem statement, and the scope of research effort. An introduction to wavelet transforms and digital watermarking is also provided.

Chapter 2 develops the discrete wavelet transform, discusses its implementation as a set of filter banks, and discusses the implementation and usage of the two-dimensional discrete wavelet transform. Both multiresolution and non-multiresolution based watermarking techniques are discussed. Compression basics are discussed to include linear and non-linear approximations as well as modeling. Finally, a review of two compression algorithms is covered: *Embedded Zerotree Wavelet Algorithm* (18) and *A Lossy Image Codec Based on Index Coding* (23).

Chapter 3 presents the design and methodology from initial audio embedding techniques using digital watermarks to a combined audio embedding audio/video compression technique. This final audio embedding audio/video compression technique combines several concepts. First, digital watermarking is used to embed the audio signal into the video signal. Second, bit-plane coding is used to apply bits to areas of the audio/video signal where they are needed most. Third, index coding is used to code the indices of the wavelet coefficients. Finally, Huffman coding is used to code the signs of the wavelet coefficients and the first-difference of their indices.

Chapter 4 presents the input audio and video data. The experiment process is covered to include the design of the experiments and the choice of watermarking, wavelet, and compression parameters. The results of the experiments are presented and discussed in detail.

Finally, Chapter 5 summarizes the final audio embedding and audio/video compression system developed. All four methods designed during this thesis effort are reviewed and compared. The results of the final audio embedding and audio/video compression system are covered. Suggested areas of future research is discussed. Finally, the contributions of this thesis effort is provided.

# *II. Background*

This chapter provides background information necessary to understand the *Methodology and Design* segment presented in *Chapter 3.* In this chapter, the peak-signal-to-noise ratio (PSNR) method of measuring image quality is discussed. General wavelet theory for the 1–$D$ wavelet transforms is provided followed by a brief discussion of biorthogonal filters and the 2–$D$ wavelet transform. Models of the human visual system are discussed in order to provide a basic understanding of how watermarking methods can take advantage of image properties to hide information. Common watermarking characteristics are presented for a basis of digital watermarking techniques. Several digital watermarking techniques are covered to provide insight into the watermark based audio insertion strategy used in this thesis. Image compression basics are provided as a precursor to the *Embedded Zerotree Wavelet Algorithm* and *A Lossy Image Codec Based on Index Coding* methods of image compression. Finally, entropy coding is discussed.

## *2.1 Dynamic Range Compression and Reverse Images*

Dynamic range compression (see Equation 2.1) is used to view those images that have been transformed via the discrete wavelet transform (DWT.) By limiting the dynamic range of the wavelet coefficients, the details of discrete wavelet transformed image are easier to see.

$$w_i' = \log_{10}\left(1 + |w_i|\right), \tag{2.1}$$

where $w_i'$ is the dynamic range-limited wavelet coefficient and $w_i$ is the original wavelet coefficient.

In some cases, the "reverse image" may be used. Viewing the reverse image can make it easier to see the contents of an image. In the reverse image, the color mapping is reversed. That is to say, the values for white and various shades of light-gray are transformed to black and corresponding shades of dark gray. To view the reverse image using the *imagesc()* command in Matlab, the negation of the original image is used.

### *2.2 Measuring Image Quality*

Many publications regarding image processing use peak-signal-to-noise-ratio (PSNR) as a measure of image quality. PSNR provides a quantitative measure (in decibels) of the similarities of two images. PSNR is defined as:

$$PSNR \;=\; 20\log_{10}\left(\frac{\max \mid x_i \mid}{\sqrt{\sum(x_i-\hat{x}_i)^2/N}}\right)$$

In the above equation, $x_i$ are the coefficients of the original image and $\hat{x}_i$ are the coefficients in the reconstructed image. It is generally accepted that if the PSNR is 30dB or greater, then the reconstructed image is considered to be a "high-likeness" of the original image.

Although other measures of image quality exist, the PSNR method is used in this thesis because it provides a quantitative measure of the similarities between two images, thus allowing the comparison of different compression qualities. Furthermore, PSNR is a common measure of image quality used throughout the image processing community.

### *2.3 Discrete Wavelet Transform*

This thesis effort investigates audio embedding and audio/video compression in the wavelet domain. It is therefore necessary to provide a general overview of the wavelet transform. Further information can be obtained from (1, 4, 10, 17, 25).

*2.3.1 Wavelet Spaces.* One of the primary themes in multiresolution analysis is the successive projection of a function $f$ into smaller orthogonal subspaces. These subspaces are formed from shifts and dilations of a lowpass scaling function $\phi(t)$ and a band pass wavelet function $\psi(t)$.

Let $\{V_m\}_{m\in\mathcal{Z}}$ be a sequence of nested subspaces in $L^2(\mathcal{Z})$. The requirement is for $V_m \subset V_{m-1} \forall m \in \mathcal{Z}$. Now let $f$ be a function in the subspace $V_{m-1}$ for some $m$. Let $P_m$ be the projection operator which takes the function $f \in V_{m-1}$ to the nested subspace $V_m \subset V_{m-1}$. The projection operator $P_m$ eliminates that part of $f$ which is not in $V_m$, yet does not disturb the portion of $f$ in $V_m$.

Let $Q_m = \mathbf{I} - P_m$ be the orthogonal projection operator that projects the function $f$ into the subspace $W_m \subset V_{m-1}$. Here, $\mathbf{I}$ is defined as the identity operator.

Because $P_m$ and $Q_m$ are orthogonal, $P_m Q_m = Q_m P_m = \mathbf{0}$. Here $\mathbf{0}$ is defined as the zero operator.

Let $\{W_m\}_{m \in \mathcal{Z}}$ be another sequence of nested subspaces such that $W_m \subset W_{m-1} \forall m \in \mathcal{Z}$. Let the span of $V_m \cup W_m = V_{m-1}$ and let $V_m$ and $W_m$ be orthogonal subspaces (i.e., $V_{m-1} = V_m \oplus W_m$ and $V_m \cap W_m = \emptyset$). Figure 2.1 illustrates the concept of these orthogonal nested subspaces.

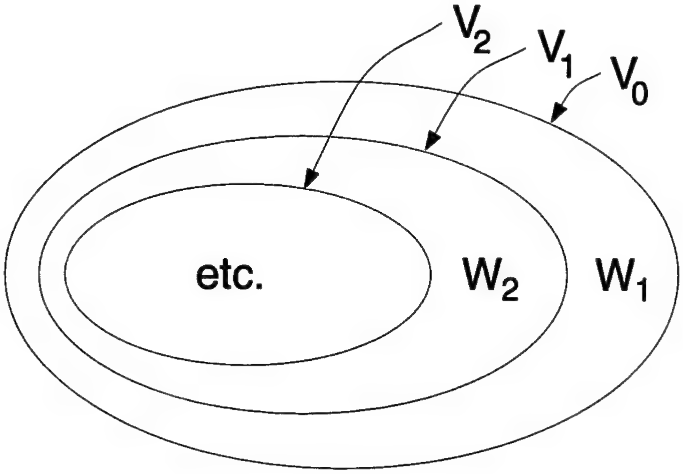


Figure 2.1. *Example of nested subspaces.*

*2.3.2 Wavelet Recursion Relation.* Let the set of functions $\{\phi_{m,l}\}_{l \in \mathcal{Z}}$ be an orthonormal basis for $V_m$ and let the set of functions $\{\psi_{m,l}\}_{l \in \mathcal{Z}}$ be an orthonormal basis for $W_m$. Because $V_m \subset V_{m-1}$, $\phi_{m,l}$ can be expanded in terms of $\phi_{m-1,k}$ as:

$$\phi_{m,l}(x) = \sum_k c_{m-1,k}\, \phi_{m-1,k}(x). \tag{2.2}$$

To determine the value of the coefficient $c_{m-1,l}(k)$, take the inner product of $\phi_{m,l}$ with the basis function $\phi_{m-1,k}$. That is:

$$\langle \phi_{m,l}, \phi_{m-1,k} \rangle = \int_{-\infty}^{\infty} \phi_{m,l}(x) \phi_{m-1,k}(x) dx \tag{2.3}$$

Let $h_{l,k} = \langle \phi_{m,l}, \phi_{m-1,k} \rangle$, then the new expansion of $\phi_{m,l}$ becomes:

$$\phi_{m,l}(x) = \sum_k h_{m-1,k}\ \phi_{m-1,k}(x). \tag{2.4}$$

In order to form an orthogonal wavelet transform, the set of functions $\phi_{m,l}$ must satisfy Equation 2.4 for all $m$. For a discrete wavelet transform (DWT), each $\phi_{m,l}$ must be constructed from integer shifts and translations ($l$ and $2^M$, respectively) of the scaling function $\phi(x)$. Therefore, Equation 2.4 can then be written as:

$$\phi(x-l) = \sum_k h(k-2l)\ \phi(2x-k). \tag{2.5}$$

In a similar fashion, the wavelet equation is written as

$$\psi(x-l) = \sum_k g(k-2l)\ \phi(2x-k). \tag{2.6}$$

In Equation 2.6, $g(k - l2) = \langle \psi_{m,l}, \phi_{m-1,k} \rangle$. Together, Equations 2.5 and 2.6 are called the *wavelet recursion relations* (4).

*2.3.3 Constructing the Discrete Wavelet Transform.* A set of orthogonal subspace is created in Section 2.3.2 in order to create a multiresolution decomposition of some signal $f$. In order for this multiresolution decomposition to work, it is assumed that the signal $f$ exists in its entirety in subspace $V_{m-1}$ where $m \in \mathcal{Z}$ (where $m$ is usually 1.) In other words, the function $f$ can be completely described as a linear combination of the basis functions in the subspace $V_{m-1}$.

The projection of $f$ into the subspace $V_m$ can be described in terms of the basis functions for $V_m$. Similarly, the projection of $f$ into the subspace $W_m$ can be described in terms of the basis functions for $W_m$. These projections of $f$ onto $V_m$ and $W_m$ are:

$$[P_m f](x) = \sum_l c_{m,l}\ \phi_{m,l}(x),$$
$$[Q_m f](x) = \sum_l d_{m,l}\ \psi_{m,l}(x),$$

where $c_{m,l} = \langle P_m f, \phi_{m,l} \rangle$ and $d_{m,l} = \langle Q_m f, \psi_{m,l} \rangle$.

Because the sum of the projection operators $P$ and $Q$ is the identity operator $I$ (i.e., $P + Q = \mathbf{I}$), $f$ can be written as:

$$f = P_m f + Q_m f. \tag{2.7}$$

Thus, any given decomposition coefficient in $V_m$ can be written as:

$$\begin{aligned} c_{m,l} &= \langle P_m f, \phi_{m,l} \rangle \\ &= \langle (f - Q_m f), \phi_{m,l} \rangle \\ &= \langle P_m f, \phi_{m,l} \rangle - \langle Q_m f, \phi_{m,l} \rangle \\ &= \langle f, \phi_{m,l} \rangle . \end{aligned} \tag{2.8}$$

If $f$ is expanded in terms of the basis functions in $V_{m-1}$ and the results of which are substituted into the expression for $c_{m,l}$, then

$$\begin{aligned} c_{m,l} &= \langle f, \phi_{m,l} \rangle \\ &= \left\langle (\sum_k c_{m-1,k}\ \phi_{m-1,k}), \phi_{m,l} \right\rangle \\ &= \sum_k c_{m-1,k}\ \langle \phi_{m-1,k}, \phi_{m,l} \rangle . \end{aligned}$$

Similarly, if $f$ is expressed in terms of the basis functions in $W_{m-1}$ then

$$d_{m,l} = \sum_k c_{m-1,k} \langle \phi_{m-1,k}, \psi_{m,l} \rangle .$$

As stated in Section 2.3.3, for the case of the discrete wavelet transform, the inner products are independent of the current decomposition level. Thus,

$$\langle \phi_{m,l}, \phi_{m-1,k} \rangle = h(k - 2l), \tag{2.9}$$

$$\langle \psi_{m,l}, \phi_{m-1,k} \rangle = g(k - 2l). \tag{2.10}$$

If the coefficients $c_{m-1,n}$ of $f$ are known, then Equations 2.11 and 2.12 give the coefficients of the projection of $f$ into $V_m$ and $W_m$ respectively. The coefficients are determined as follows:

$$c_{m,l} = \sum_k c_{m-1,k}\, h(k-2l), \tag{2.11}$$

$$d_{m,l} = \sum_k c_{m-1,k}\, g(k-2l). \tag{2.12}$$

Equations 2.11 and 2.12 lead to a filter bank implementation of the discrete wavelet transform as depicted in Figure 2.2.

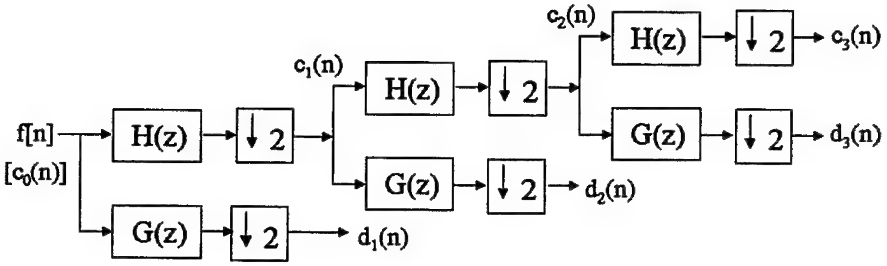


Figure 2.2. *Filter bank implementation of the discrete wavelet transform.*

Once $c_{m,l}$ and $d_{m,l}$ are known, the function $f$ can be decomposed onto the orthogonal subspaces of $V_m$. The orthogonal subspaces of $V_m$ are $V_{m+1}$ and $W_{m+1}$. The subspaces of $V_m$ can be decomposed using the same formulas because the filters $h$ and $g$ are independent of the decomposition level $m$ and lead to a recursive decomposition (i.e., the DWT decomposition is multiscale.) That is, the DWT consists of a set of *scaling coefficients* $c_m[n]$, which represent coarse signal information at some scale $m = M$, and a set of wavelet coefficients $d_m[n]$, which represent detail signal information at scales $m = 1, 2, \ldots, M$.

*2.3.4 Reconstruction.* The previous sections demonstrate how a given signal $f$ can be decomposed into a set of scaling coefficients and a set of wavelet coefficients. It is now necessary to show how $f$ can be reconstructed using $c_{0,k}$ and $d_{0,k}$.

The signal $f$ exists in $V_0$ for scale $m = 1$. It is represented in $V_0$ as $f(x) = \sum_k c_{0,k}\ \phi_{0,k}(x)$. From the decomposition of $f$ into the spaces $V_0$ and $W_0$, we have the scaling and wavelet coefficients ($c$ and $d$ respectively.) $V_0$ was decomposed into a set of orthogonal subspace $V_1$ and $W_1$ so $f(x)$ becomes:

$$f(x) = \sum_k c_{1,k}\ \phi_{1,k}(x) + \sum_k d_{1,k}\ \psi_{1,k}(x). \tag{2.13}$$

If the inner product is taken on both sides of Equation 2.13 with $\phi_{0,n}(x)$, $\langle f, \phi_{0,n}\rangle = c_{0,n}$. The right side of Equation 2.13 is reduced to $c_{0,n}$ because the $\{\phi_{0,n}\}$ are orthonormal and the subspaces $V_0$ and $W_0$ are orthogonal. Recall Equations 2.9 and 2.10 (Section 2.3.3):

$$\begin{aligned}\langle \phi_{m,l}, \phi_{m-1,k}\rangle &= h(k-2l),\\ \langle \psi_{m,l}, \phi_{m-1,k}\rangle &= g(k-2l).\end{aligned}$$

They can be substituted into Equation 2.13 to yield

$$c_{0,n} = \sum_k c_{1,k}\ h(n-2k) + \sum_k d_{1,k}\ g(n-2k). \tag{2.14}$$

Equation 2.14 leads to the filter bank implementation of the inverse DWT. In the orthogonal DWT, the lowpass filter $h$ and the highpass filter $g$ used in the inverse DWT are identical to the lowpass and highpass filters used in the forward DWT.

### *2.4 Biorthogonal Discrete Wavelet Transform*

By loosening the orthogonality requirement, a biorthogonal wavelet transform can be created (4, 25). For image processing, biorthogonal filters are generally more desirable than orthogonal filters for two reasons. First, it is easy to design biorthogonal filters that have linear phase. Filters with linear phase must be symmetric, which implies that they must be odd length filters (with the exception of the Haar filters.) All orthogonal wavelets are the result of even length filters, are not symmetric, and thus do not have linear phase (again, with the exception of the Haar filters.) Maintaining the integrity of phase information within an image is important because phase information is much more significant than magnitude information in image reconstruction (12) (see Appendix B for further information on the importance of phase information.) Second, biorthogonal wavelets allow for larger filter sizes. In general, larger size filters correspond to a smoother signal representation. Details of the development of the biorthogonal wavelet transform can be found in Appendix A.1.

### *2.5 Two Dimensional Wavelet Transform*

The previous sections describe the wavelet transform theory for 1–*D* signals. This section discusses the application of the wavelet transform for images.

Much like the 2–*D* Discrete Fourier transform, the 2–*D* discrete wavelet transform is a separable transform. That is, to perform the transform in two dimensions, the filters are applied along the rows and then the columns. The discrete wavelet transform is a dyadic decomposition of the original two dimensional signal (an image), where each of the four signal approximations is referred to as a subband. Figure 2.3 displays the original "Lenna" image. Figure 2.4 illustrates the subbands generated after one iteration of the DWT. Each subband is unique. The HH subband contains those elements that are highpass filtered along the rows and then highpass filtered along the columns. It preserves edges (which are the high-frequency portions of an image) at 45° angles thus creating a "cross-hatch" effect. The HL and LH subbands are commonly referred to as the bandpass subbands. The HL subband corresponds to highpass filtering along the rows and lowpass filtering along the columns and tends to preserve horizontal edges. The LH subband corresponds to lowpass filtering along the rows and highpass filtering along the columns and tends to preserve vertical edges. The HL, LH, and HH subbands are collectively called the detail subbands. Finally, the LL subband corresponds to lowpass filtering the rows and then lowpass filtering the columns and thus tends to smooth out the edges in an image. The LL subband is considered the coarse approximation to the original image. It should be clear that the DWT provides a coarse approximation to the image (the LL subband) with a series of details (the HH, HL, and LH subbands.)

Lower scales are created with each iteration of the DWT. Each additional iteration of the DWT processes the coarse approximation (LL subband), each time creating a new dyadic decomposition with each of the four subbands (LL, HL, LH, and HH.) Figure 2.5 illustrates the multiscale effect of the discrete wavelet transform.

Figure 2.3. *Original* 256 × 256 *pixel* 8-*bit grayscale "Lenna" image.*

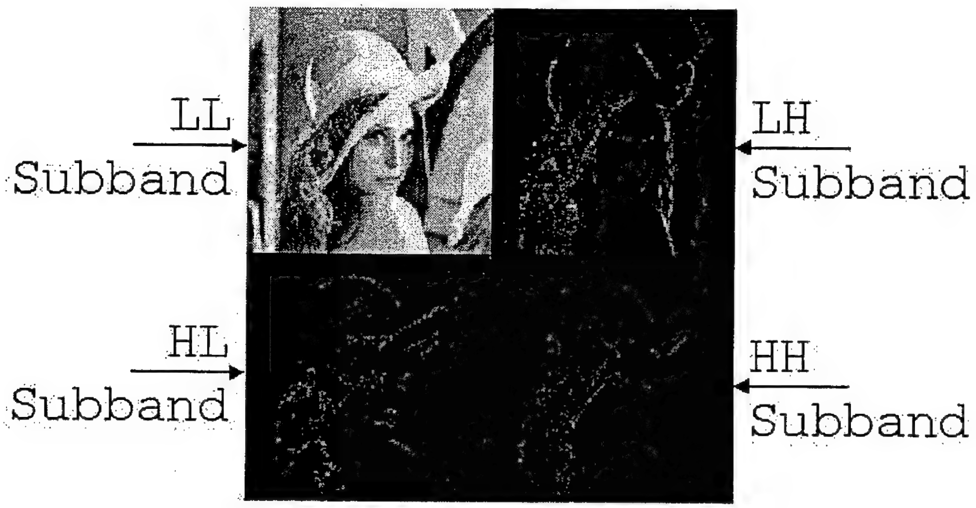


Figure 2.4. *The original "Lenna" image after one iteration of the Discrete Wavelet Transform using Daubechies* (7, 9) *biorthogonal filters. This dyadic decomposition of the "Lenna" image has four distinct subbands. The HH subband preserves edges in the* 45° *directions, the HL subband preserves horizontal edges, the LH subband preserves vertical edges, and the LL subband provides a coarse approximation to the original image. This image is displayed using dynamic range compression.*

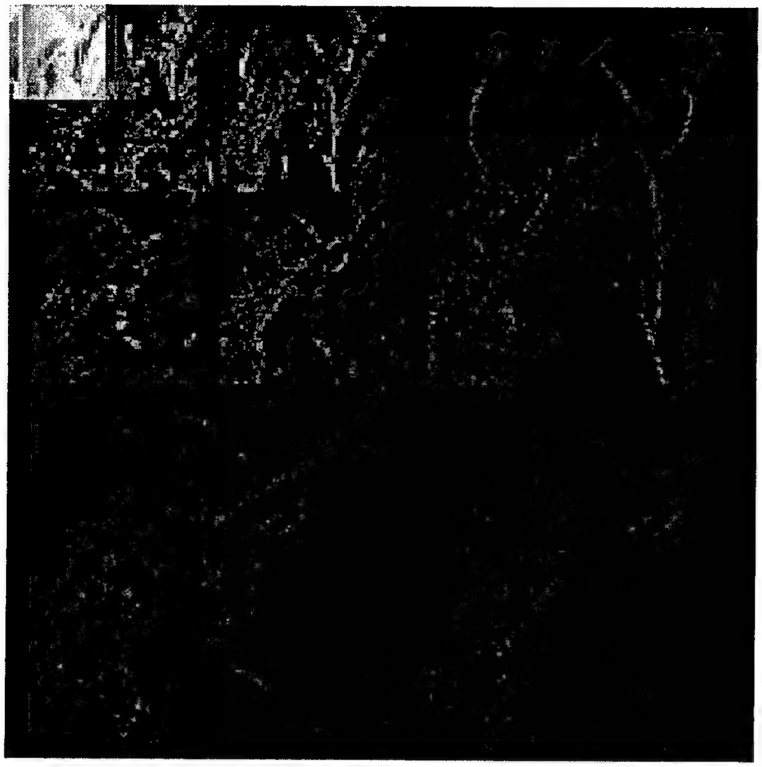


Figure 2.5. *The original" Lenna" image after three iteration of the Discrete Wavelet Transform using Daubechies* $(7, 9)$ *biorthogonal filters. This example clearly illustrated the multiscale effect. Notice that the LL subband has a likeness to the original "Lenna" image, where the subbands at different scales provide details within the image. This image is displayed using dynamic range compression.*

## *2.6 Human Perception*

Vision models describe the characteristics of the human visual system (HVS) and are used to design image processing systems such as transforms, compression algorithms, and watermarking techniques. Visual masking is the ***inability*** of the HVS to see certain aspects of an image due to image properties. In other words, masking is the ability to hide noise or data in different regions of an image. These regions are generally located around edges and textured areas. Visual masking phenomena are:

1. Luminance masking – Bright background regions mask embedded signals with high magnitude.
2. Frequency masking – Weak signals at one frequency are masked by strong signals at a nearby frequency.
3. Spatial masking – Spatial patterns reduce the visibility of neighboring areas (22).
4. Texture masking – Strong background texture decreases the visibility of hidden objects (8).

The just noticeable difference (JND) determines an upper bound (or threshold) at which noise levels below this upper bound are imperceptible. For images, the JND depends on the masking effects, light sensitivity, and spatial frequency sensitivity (27).

Sensitivity of the HVS to spatial frequency is modeled using the modulation transfer function (MTF.) The MTF describes the tolerance of the HVS to noise levels at different spatial frequencies (27). Light sensitivity is the dependence on local luminance. For an 8-bit grayscale image, this means that high visibility thresholds generally appear in light regions (those values close to 255) or dark regions (those values close to 0) and low visibility thresholds appear in mid-grayscale regions (those values close to 127.)

## *2.7 Embedded Audio/Video*

Relatively little research has been performed on techniques to embed audio information into an image. However, one such technique for embedding audio information into an image is presented in (29). The audio information is not truly "embedded" within the original image. Rather, it is added to the bottom of the original image as an audio image. An audio image is created by taking the one-dimensional audio signal (a vector) and transforming it into a two-dimensional signal (an image.) The $M \times N$ two-dimensional signal is created by taking the first $M$ coefficients of the one-dimensional audio signal as the first row of the two-dimensional audio image. The second set of $M$ coefficients of the one-dimensional audio signal are used as the second row of the two-dimensional audio image. This processes is iterated $N$ times, resulting in the $M \times N$ audio image. To extract the audio information, the rows of the audio image are combined to create a one-dimensional vector, which is now the original audio signal.

## *2.8 Digital Watermarks*

Watermarks are generally thought of as patterns on pieces of paper. These patterns are visible when the paper is held up to a light source. Watermarks were used as early as the 13th century to identify their particular makers and also to prove authenticity of paper money (6). A digital watermark is an image or data stream that is embedded into digital media. The medium could be a digital image, a digital audio stream, or even a digital video stream. The main focus of current digital watermarking research is copyright protection of digital media.

Two categories of digital watermarking techniques are discussed below: non-multiresolution techniques and multiresolution techniques. The non-multiresolution techniques use the discrete cosine transform (DCT.) A majority of the multiresolution techniques utilize the discrete wavelet transform (DWT.)

Throughout the discussion on watermarking methodologies, the "receiving image" refers to the image that receives the watermark information; it has specific coefficients modified in some manner.

*2.8.1 Watermarking Characteristics.* The literature presents several characteristics of watermarking techniques. The two primary characteristics are imperceptibility and robustness:

1. Imperceptibility – The embedded watermark should not degrade image quality. Image quality is generally determined via peak-signal-to-noise ratio (PSNR.) Typically, a PSNR of 30dB or greater is considered to be a high-quality version of the original image.

2. Robustness – The watermark should be inserted in such a way that it survives common signal processing functions. Common signal processing functions include: image compression, linear or nonlinear filtering such as Wiener filtering for image restoration, image enhancement techniques such as contrast modification and histogram equalization, sub-sampling, quantization, digital-to-analog conversion, and analog-to-digital conversion. The watermarking technique should also be immune to variations in image rotation, translation, and scale (3). This research concentrates on robustness against compression.

These two primary characteristics conflict which makes it necessary to design the watermarking technique in order to obtain a balance between the two.

Secondary characteristics are those watermarking characteristics which are desirable. These secondary characteristics are not necessary, and are not present in many watermarking techniques. Their importance is driven by the current application. The secondary characteristics are presented below:

1. Unambiguity – The watermark should be unique and identify the owner of the medium (3, 28).

2. Real-time processing – The watermarking technique should allow for real-time insertion and extraction (28).

3. Extraction without original information – The watermarking technique should allow the extraction of the watermark without using the original image or the original watermark (28).

4. Universality – The watermarking technique should work for images, video, and audio on common hardware (3).

For the purposes of this research, item 3 above (*extraction without original information*) is the only secondary characteristic that is of significant importance. However, item 2 (*real-time processing*) is considered throughout the development of the audio embedding process.

### *2.8.2 Watermarking Methodology.*

According to Cox et al (3), a watermarking technique contains two major components: watermark structure and insertion strategy. In this thesis, the watermark structure is a binary audio signal. The watermark insertion strategy satisfies the primary and secondary characteristics as discussed in Section 2.8.1. The specific watermarking technique embeds a binary audio signal into a series of digital images (where each image is a frame in a video sequence) and extracts the inserted watermark without the need for significant overhead.

Four watermarking insertion domains are considered: spatial domain, Fourier domain, discrete cosine domain, and wavelet domain. The non-spatial domain techniques dominate watermarking research. Few spatial domain techniques have been developed because they are not as robust as the other techniques. This research uses a wavelet domain technique for the watermark insertion strategy.

### *2.8.3 Non-Multiresolution techniques.*

#### *Adaptive Watermarking.*

The DCT-based watermarking technique, developed by Huang et al (8), is based on two visual masking constructs: luminance masking and texture masking. The digital watermark is a randomly generated number sequence. The watermark embedding process works as follows:

1. Transform the watermark receiving image into $k$ $8 \times 8$ non-overlapping blocks.
2. Perform the DCT on each $8 \times 8$ block.
3. Assign each $8 \times 8$ block to one of the three following classes based on the estimation of the average brightness and texture complexity: class 1, dark with weak texture; class 2, bright with strong texture; and class 3, other.
4. Modify the DCT coefficients with the watermark information.

(a) Divide the watermark coefficients into groups, each group with three consecutive values from the watermark.

(b) Insert each group of watermark coefficients (see Equation 2.15) into one of the $k$ $8 \times 8$ blocks according to the following rules:

i. If the current block is in Class 1, then the three watermark coefficients are inserted into the significant DCT coefficients of that block using an $\alpha_k = 2$.

ii. If the current block is in Class 2, then the three watermark coefficients are inserted into the significant DCT coefficients of that block using an $\alpha_k = 6$.

iii. If the current block is in Class 3, then the three watermark coefficients are inserted into the significant DCT coefficients of that block using an $\alpha_k = 9$.

5. Perform the inverse DCT on each $8 \times 8$ block.

6. Reassemble the $k$ $8 \times 8$ blocks to generate the watermarked image.

Equation 2.15 describes the insertion of the watermark information into the DCT coefficients.

$$F'_k(u,v) = \begin{cases} F_k(u,v) + \alpha_k x_i & 3k \leq i < 3(k+1) \\ & (u,v) \in \{(0,1),(1,0),(1,1)\} \\ F_k(u,v) & \text{Otherwise} \end{cases} \tag{2.15}$$

where:

1. $(u,v)$ are the locations of the DCT coefficients that receive the watermark information.

2. $F'_k(u,v)$ are the modified DCT coefficients in block $k$.

3. $F_k(u,v)$ are the original DCT coefficients in block $k$.

4. $\alpha_k$ is the scaling factor ($\alpha_k = 2, 6, 9$ for Class $1, 2, 3$ respectively.)

5. $3k \leq i < 3(k+1)$ indexes the three watermark coefficients that are embedded into the DCT coefficients in block $k$.

6. $x_i$ are the watermark coefficients.

The correlation function is used to determine the similarity between the extracted watermark and the original watermark. The correlation function is defined as:

$$corr(w', w) = \frac{\sum_i x_i' x_i}{\sum_i x_i'^2} \tag{2.16}$$

where:

1. $w$ is the original watermark.
2. $w'$ is the extracted sequence.
3. $x_i$ is the original $i^{th}$ watermark coefficient.
4. $x_i'$ is the extracted $i^{th}$ watermark coefficient.

Generally, a correlation of six (i.e., six standard deviations) indicates that the probability of incorrectly identifying the correct watermark is very small.

*Grayscale Watermarking.* Most watermarking techniques use a binary image or a binary data sequence as the watermark. A technique for embedding grayscale images as watermarks is presented by Niu et al (16). In this technique, the watermark is separated into $L$ bit-planes. Here, $L$ is determined by:

$$L = \log_2(number\_of\_gray\_levels).$$

Each of the $L$ bit-planes is a binary image. Figure 2.6 illustrates this bit-planning method.

The watermark insertion process works as follows:

1. Transform the watermark receiving image into $N$ $8 \times 8$ blocks.
2. Perform the DCT on each $8 \times 8$ block.
3. Quantize the DCT coefficients.
4. Decompose the watermark into $L$ bit-planes as described above.
5. Embed a portion of the $L$ binary watermarks into the middle frequency components of the receiving image.

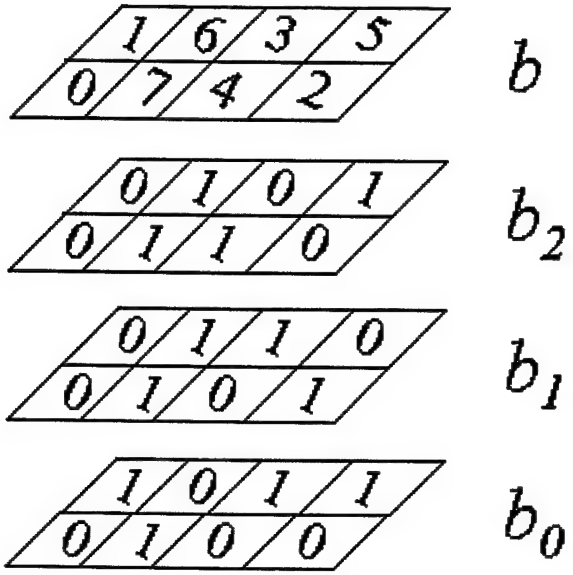


Figure 2.6. *$L$ bit planes of a grayscale image where $L = 3$.*

6. Use the remainder of the $L$ binary watermarks to generate a secret key and masking information.
7. De-quantize the watermarked image.
8. Perform the inverse DCT on each $8 \times 8$ block.
9. Reassemble the $N$ $8 \times 8$ blocks to generated the watermarked image.

To extract the watermark, the original image and masking information are used. To reconstruct the watermark, the secret key is used with the results of the extraction process. The normalized cross-correlation function (see Equation 2.16) is used to measure the similarities between the original watermark and the reconstructed watermark.

### *2.8.4 Multiresolution Techniques.*

*Torus Automorphisms and the Wavelet Transform.* The technique of digital watermarking developed by Tsai et al (24) uses the DWT on the receiving image and the Toral transform, a spatial transform, on the watermark. Tsai's work is based on the work by Voyatzis et al (26). The difference is that the original algorithm developed by

Voyatzis et al watermarks in the spatial domain. That is, the Toral transformed watermark is inserted directly into the receiving image at some point $(p_1, p_2)$.

The Toral transform is based on torus automorphisms and is a semi-chaotic transform that is periodic in nature. It is used to redistribute the energy of the watermark over some region (perhaps the same size or larger.) By distributing the energy semi-chaotically, the transformed watermark appears as noise and hence, the embedding process does not leave artifacts. The Toral transform relocates the points of the original $M_1 \times M_2$ watermark to some new location in an $N \times N$ area. The Toral transform is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ k & (k+1) \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} \bmod N \tag{2.17}$$

In Equation 2.17, $k$ is referred to as the controlling parameter (as it is the variable that controls where in the $N \times N$ array that the point is relocated), $N$ is the dimension of the resulting 2–$D$ array, $(x, y)$ is the pixel location before the transform, and $(x', y')$ is the pixel location after the transform. If the transform is applied $P$ times (where $P$ is the period of the watermark), then the original watermark appears in the upper left hand corner of the $N \times N$ area and returns to its original size, $(M_1 \times M_2)$. The period is a function of the controlling parameter $k$ and the resulting size of the transformed watermark $N$. That is, $P = \mathcal{F}(k, N)$. Figure 2.8 illustrates the Toral transform of the boy image in Figure 2.7. The dimensions of the boy image is 128 × 128 and the Toral transformed boy image is 256 × 256 ($N = 256$ and $k = 15$.) For the Toral transformed boy image in Figure 2.8, the period $P = \mathcal{F}(k, N) = 48$ and is illustrated in Figure 2.9 (the value for $P$ was determined empirically as the literature does not provide a closed form expression for determining $P$.) The Toral transformed watermark is inserted into an $N \times N$ area in the receiving image starting at some point $(p_1, p_2)$. Physical dimensions of the Toral transformed watermark are flexible and can be specified to fit in a particular subband of the receiving image. The authors chose to insert the watermark into the low-high ($LH$) region of subband one. No justification was provided as to why this subband was chosen over other subbands. The technique of watermark embedding and extraction is based on a modular threshold scheme. Nine integer values and the watermarked image are needed to

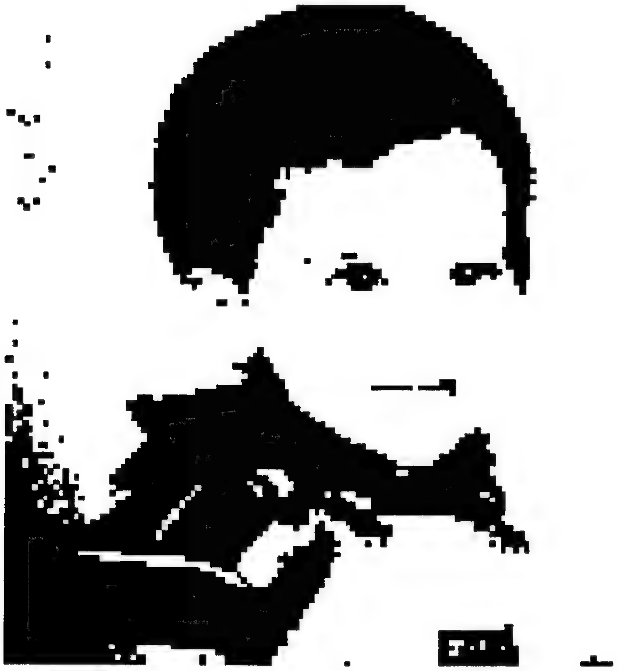


Figure 2.7. *Original* 128 × 128 *pixel* 1-*bit grayscale "Boy" image.*

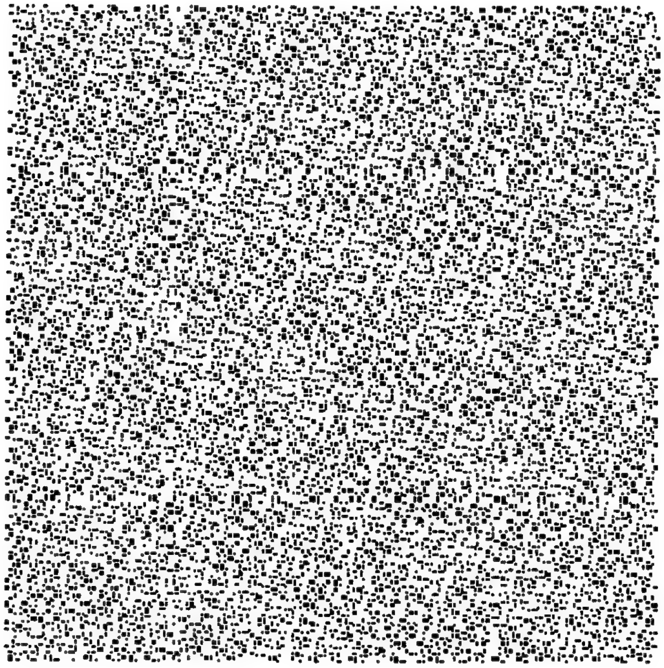


Figure 2.8. *Toral transform of the "Boy" image. The Toral transform distributes the energy of the original "Boy" image over a* 256 × 256 *pixel region. The parameters for this Toral transform are:* $N = 256$, $n = 21$, *and* $k = 15$. *This image is the reverse image.*

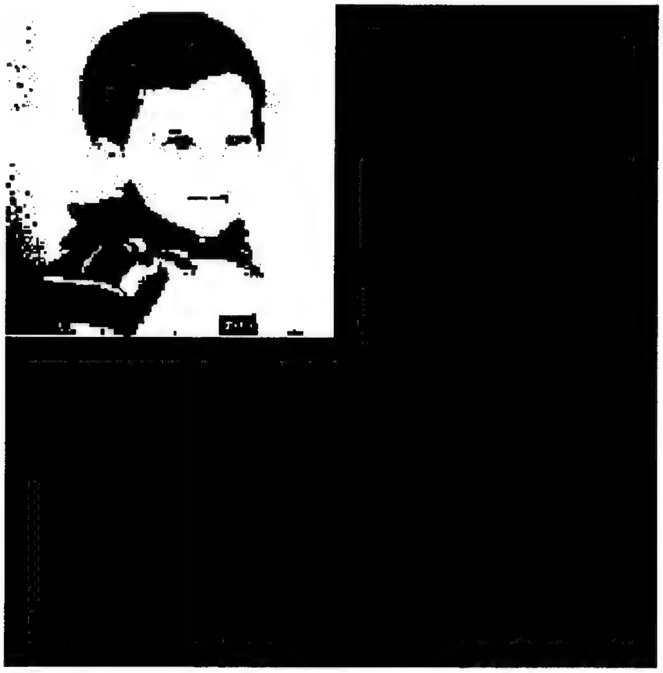


Figure 2.9. *Results after performing one period of the Toral transform. In this example, the period* $P = 48$. *The original* 128 × 128 1*-bit grayscale "Boy" image is perfectly reconstructed in the upper left corner.*

extract the embedded watermark. Watermark extraction is not dependent on the original watermark or the original image.

*Watermarking using the Just Noticeable Difference Model.* The watermarking technique developed by Wei et all (27) inserts the watermark into the wavelet coefficients based on the just-noticeable difference (JND) model (see Section 2.6.)

The watermark is inserted such that the induced noise is less than the just-noticeable difference of the receiving wavelet coefficients. Watermark insertion order is from low frequency subbands to high frequency subbands. This technique enhances the preservation of the watermark during lossy compression and low-pass filtering, enhances the robustness of the watermark, and guarantees that the watermark is transparent. Watermark extraction is based on "direct sequence spread spectrum communication and correlation detection" (27). For further information on direct sequence spread spectrum techniques, please see (3, 20).

*Adaptive Watermarking using Quantization Models.* An adaptive watermarking technique based on the quantization model of the human visual system (HVS) was developed by Kaewkamnerd et al (11). Frequency, luminance, and texture are used to calculate a weighting function at resolution level $r = 0, 1, 2, 3$ and orientation $s = \{LL,\ LH,\ HL,\ LL\}$. The weighting function is:

$$T(r, s, x, y) \quad = \quad f(r, s)l(r, x, y)t(r, x, y)^{0.034},$$

where $f$ is the frequency function, $l$ is the luminance function, and $t$ is the texture function. The functions $f$, $l$, and $t$ are defined as:

$$f(r, s) \quad = \quad \left\{ \begin{array}{ll} \sqrt{2} & \text{if } s = HH \\ 1 & \text{otherwise} \end{array} \right\} * \left\{ \begin{array}{ll} 1.00 & \text{if } r = 0 \\ 0.32 & \text{if } r = 1 \\ 0.16 & \text{if } r = 2 \\ 0.10 & \text{if } r = 3 \end{array} \right\} \qquad (2.18)$$

$$l(r, x, y) \quad = \quad 3 + \frac{1}{256} \sum_{i}^{1} \sum_{j=0}^{1} I^{3,LL}(i + 1 + x/2^{3-r}, j + 1 + y/2^{3-r}) \qquad (2.19)$$

$$\begin{aligned} t(r, x, y) \quad = \quad & \sum_{k=1}^{3-r} \sum_{s}^{(HH,HL,LH)} \sum_{i=0}^{1} \sum_{j=0}^{1} (I^{k+r,s}(i + x/2^{k} + y/2^{k}))^{2} \qquad (2.20) \\ & +16^{3-r} var(I^{3,LL}(\{1, 2\} + x/2^{3-r}, \{1, 2\} + y/2^{3-r})). \end{aligned}$$

The weighting function is used to maximize the perceptual masking of the watermark coefficients. The quantization model is used to insert the watermark into the high level subband of the receiving image. Hence, imperceptibility of the watermark is increased and the robustness of the watermark is maintained. In this technique, the watermark is a Gaussian pseudo-random sequence with zero mean and unit variance ($N(0, 1)$.) Equation

2.21 governs the insertion of the watermark into the receiving image.

$$I^{(r,s)*}(x,y) = \begin{cases} I^{(r,s)}(x,y) + \alpha(T(r,s,x,y) * X(x,y)) & \text{If } I^{(r,s)}(x,y) > \frac{T(r,s,x,y)}{4} \\ I^{(r,s)}(x,y) & \text{Otherwise} \end{cases} \tag{2.21}$$

In Equation 2.21,

1. $I^{(r,s)*}(x,y)$ is the watermark coefficient at level $r$, orientation $s$, and position $(x,y)$,
2. $I^{(r,s)}(x,y)$ is the corresponding image coefficient,
3. $\alpha$ is a scaling factor,
4. $T(r,s,x,y)$ is the weighting factor described above,
5. $X(x,y)$ is the watermark.

Using the original watermark and the extracted sequence, the similarity function is calculated to determine the correlation between the original watermark and the extracted watermark. The similarity function is the same as the correlation function in Equation 2.16. As with the other techniques that correlate the extracted watermark and the original watermark, a correlation value of six or greater indicates that the probability of incorrectly identifying the watermark is very small.

*Multiresolution Decomposition using the Resolution Reduction Function.* A multiresolution watermarking technique based on the DWT of the receiving image and the resolution-reduction function of a binary watermark was developed by Hsu et al (7). The resolution-reduction described in (5) decomposes an image into a multiresolution structure. Each of the decomposed layers of the watermark are embedded into the corresponding subbands of the wavelet transformed image.

The watermarking process is as follows:

1. Perform the DWT on the watermark receiving image.
2. Decompose the watermark using the resolution-reduction function described in (5).
3. Generate a pseudo-random permutation to disperse the decomposed watermark. The watermark appears as noise and is thus statistically undetectable.

4. Generate a block-based permutation of the binary watermark. This processes is image dependent and guarantees the imperceptibility of the watermark within the receiving image.

5. Modify the wavelet coefficients of the receiving image. The watermark is inserted in the adjacent ($LH_j$ and $HL_j$ where $j$ is the scale) "neighboring relationships" (7) of the wavelet transformed image using a residual mask.

6. Perform the inverse DWT to generated the watermarked image.

The original image and the watermarked image are used to obtain the permutation mapping from step four. The correlation between the original watermark and the extracted watermark is calculated for verification (see Equation 2.16 for the correlation function.)

*Adaptive Nonlinear Watermarking.* A watermarking technique used for images and video was developed by Zhu et al (30). The watermark is a Gaussian distributed random vector with zero mean and unit variance ($N(0,1)$). The watermark is added to the highpass subbands of the transformed image via an adaptive nonlinear insertion procedure presented in Equation 2.22 (2) and depicted in Figure 2.10 (2).

$$v_i' \quad = \quad v_i(1 + \alpha_i x_i) \tag{2.22}$$

In the above equation, $v_i'$ is the modified wavelet coefficient, $v_i$ is the original wavelet coefficient, and $x_i$ is the watermark coefficient. If the wavelet coefficient of the receiving image is small, then so is the energy of the inserted watermark coefficient. If the wavelet coefficient of the receiving image is large, then the energy of the inserted watermark coefficient is as well. The energy is controlled by the scaling coefficient $\alpha_i$. The advantage of this insertion strategy is that high energy information from the watermark is not inserted into low energy wavelet coefficients which helps alleviate artifacts in the resulting watermarked image. Additionally, by inserting low energy watermark components in low energy areas of the image, and high energy watermark components in high energy areas of the image, watermark robustness is enhanced. As with many other watermark extraction techniques,

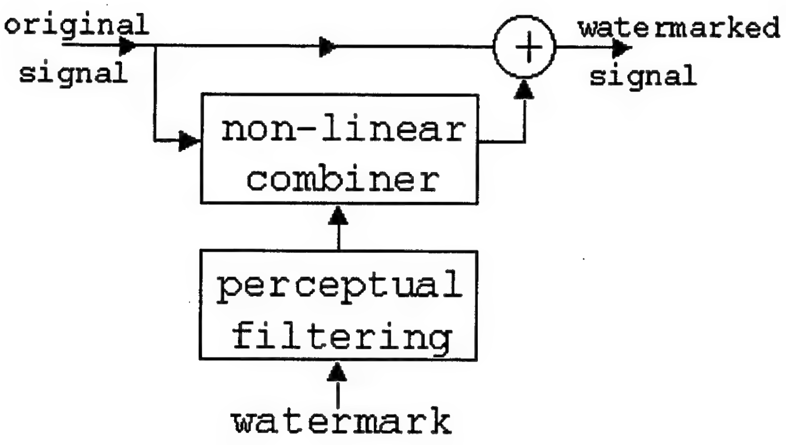
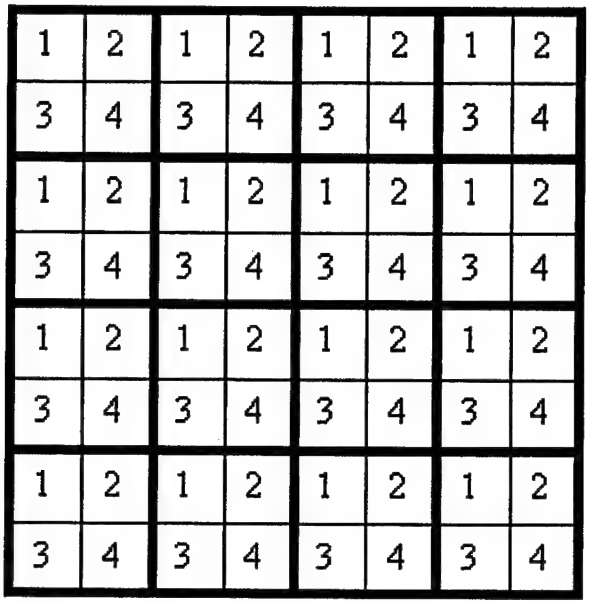


Figure 2.10. *The watermark and image are combined by filtering the watermark and using a nonlinear combiner. The filtering is accomplished by using the scaling parameter $\alpha_i$. The nonlinear combiner is simply the $v\alpha_i x_i$ term in Equation 2.22. And finally, the results of the nonlinear combiner are summed with the original wavelet coefficient $v$ to create the watermarked coefficient $v'$.*

the original watermark is compared with the extracted watermark using the correlation function in Equation 2.16.

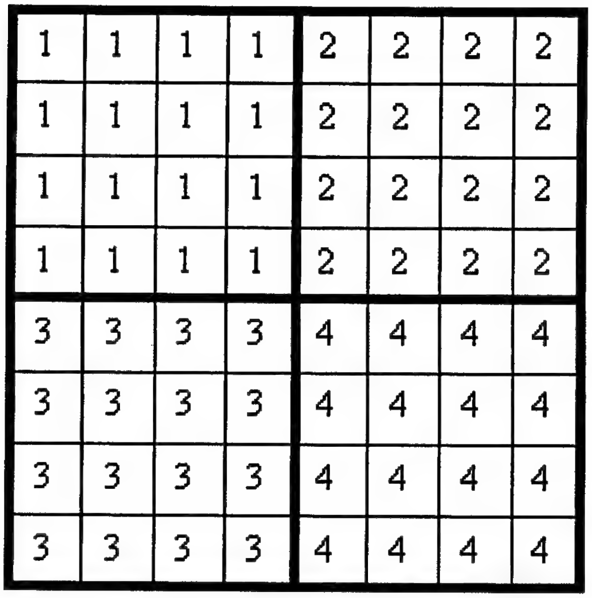
*Zerotree Structure to the Cosine Transform.* All of the multiscale watermarking techniques thus far have focused on the DWT of the receiving image. Wu et al (28) apply a wavelet structure to the DCT to show a relationship between the rearranged DCT coefficients and the resulting zerotree structure. Figures 2.11 and 2.12 show the relationships between the DCT and the DWT. Furthermore, Figures 2.13 and 2.14 illustrate the coefficient indices of the DCT coefficients and the rearranged DCT coefficients.

As with many of the wavelet watermarking techniques, this DCT based technique inserts the watermark into visually significant portions of the receiving image. Therefore, the watermark is robust and can withstand lossy compression and general signal processing functions. Inserting the watermark into visually significant portions of the image also ensures that the watermark does not have significant impact on image fidelity. To insert the watermark into the visually significant portions of the spectrum, the frequency domain representation of the image (i.e., the transformed image) is viewed as a communications channel and the watermark is viewed as the signal. In doing so, the watermark is dis-



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 |

Figure 2.11. *Structure of the discrete cosine transformed image with* 2 × 2 *non-overlapping blocks.*



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 |
| 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 |
| 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 |
| 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 |

Figure 2.12. *Coefficients of* 1 *iteration of the discrete wavelet transformed image.*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 6 | 7 | 15 | 16 | 28 | 29 |
| 3 | 5 | 8 | 14 | 17 | 27 | 30 | 43 |
| 4 | 9 | 13 | 18 | 26 | 31 | 42 | 44 |
| 10 | 12 | 19 | 25 | 32 | 41 | 45 | 54 |
| 11 | 20 | 24 | 33 | 40 | 46 | 53 | 55 |
| 21 | 23 | 34 | 39 | 47 | 52 | 56 | 61 |
| 22 | 35 | 38 | 48 | 51 | 57 | 60 | 62 |
| 36 | 37 | 49 | 50 | 58 | 59 | 63 | 64 |

Figure 2.13. *Original coefficients of the discrete cosine transformed image.*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 5 | 6 | 17 | 18 | 21 | 22 |
| 3 | 4 | 7 | 8 | 19 | 20 | 23 | 24 |
| 9 | 10 | 13 | 14 | 25 | 26 | 29 | 30 |
| 11 | 12 | 15 | 16 | 27 | 28 | 31 | 32 |
| 33 | 34 | 37 | 38 | 49 | 50 | 53 | 54 |
| 35 | 36 | 39 | 40 | 51 | 52 | 55 | 56 |
| 41 | 42 | 45 | 46 | 57 | 58 | 61 | 62 |
| 43 | 44 | 47 | 48 | 59 | 60 | 63 | 64 |

Figure 2.14. *Rearranged coefficients of the discrete cosine transformed image.*

tributed over many frequency bins. Each bin has a small portion of the energy contained in the watermark, which is the key to the robustness of this technique. To extract the watermark, neither the original image nor the original watermark are needed.

*Discussion of Watermarking Methodologies.* There are several watermark extraction themes presented in the above watermarking methods: correlation of extracted watermark with original watermark, extraction of watermark using the original image, direct sequence spread spectrum techniques, and watermark extraction without original information. The watermarking techniques have one common thread. That is, they insert the watermark in visually significant regions of the image in order to maintain a balance between the two primary watermarking characteristics: robustness and imperceptibility.

Although many of the methods presented above extract the watermark without the use of original information (8, 24, 27, 11, 30, 28), many require a correlation between the original watermark and the extracted watermark (8, 11, 30). The correlation is required because these techniques use random or pseudo random (noise) sequences as the watermark. These correlation based techniques may not work well for the audio embedding application because even with detrimental audio bit errors, a correlation of six between two signals can potentially be attained.

The watermarking technique developed by Tsai et al (24) is used in this research because it requires very little overhead to extract the watermark which coincides with the "extraction without original information" secondary characteristic. Additionally, the watermark embedding/extracting process consists of relatively simple mathematical operations which potentially lends this technique to real-time processing applications.

## *2.9 Compression*

A few basic concepts regarding compression is covered before discussing the two compression methods investigated during this research: *Embedded Zerotree Wavelet Algorithm* (18) and *A Lossy Image Codec Based on Index Coding* (23).

The first concept is **linear approximation**. A linear approximation, in general, is an approximation that would always yield the same coefficients in every image. For example, a linear approximation to an image might be to keep the first $N$ wavelet coefficients. For Fourier analysis, linear approximation is well suited because the most important signal information lies within the first $N$ coefficients of the the Fourier series representation. There is a distinct advantage to the linear approximation: there is no overhead involved in maintaining position information. However, in the wavelet domain it is very unlikely that the first $N$ coefficients contain enough of the right information to create a high-quality reconstruction of the original image due to the space localization property of the wavelet transform. This leads to an investigation of nonlinear approximation.

The concept of **nonlinear approximation**, in general, is an approximation that is image dependent. That is, an approximation for one image is very unlikely to be an approximation to a different image. An example of nonlinear approximation is to keep the $N$ largest magnitude wavelet coefficients. The disadvantage of this nonlinear approximation is that there is a fair amount of overhead involved in maintaining the locations of those $N$ coefficients. However, the advantage is that most of the visually significant information is contained in the largest wavelet coefficients. Furthermore, the wavelet coefficients tend to decay rapidly, leaving only a handful of large wavelet coefficients; thus, wavelets are well suited for nonlinear approximation applications such as image compression.

Third, a **model** is often helpful when developing a compression algorithm. For wavelet-based compression, it is necessary to create a model that exploits the wavelet structure. If the model is accurate, then the position information does not need to be stored because the model allows for the reproduction of the image without it. If the model is inaccurate, then poor compression rates result.

Fourth, an efficient and effective method of **coding** coefficients is important. Bit allocation is often desired in order to apply bits to regions of the image that need them most. This is often accomplished by bit plane coding, and it is used in many compression algorithms. Bit plane coding consists of two aspects: initial bit assignment and bit refinement. When a new value is located, it is given an initial bit, either a 1 or a 0 depending on its value and the threshold value being compared against. After a coefficient has been given an initial bit, it is refined on all subsequent passes (however many there may be.)

To illustrate the concept of bit plane coding, if the current comparison value (or threshold) $T = 64$, then the coefficients from $64 \leq w_i \leq 128$ are analyzed. If the current wavelet coefficient, $w_i$, is larger than 64, it is assigned a 0 if it less than 96 or a 1 if it is greater than or equal to 96 ($96 = \frac{3*T}{2}$.) If the value is assigned a 0, then its reconstruction value is 80 ($80 = \frac{5*T}{4}$.) If the value is assigned a 1, then it is reconstructed as 112 ($48 = \frac{7*T}{4}$.)

On each iteration of the bit plane coding method, the value of $T$ is reduced by a factor of two, refinement bits are assigned to previous significant values, and new significant values are assigned an initial bit. The reconstruction value of a coefficient is accurate within $T/4$. The more iterations that occur, the more $T$ is reduced, and the more accurate the reconstructed values become.

*2.9.1 Embedded Zerotree Wavelet Algorithm.* The *Embedded Zerotree Wavelet Algorithm* was developed by Jerome M. Shapiro (18). It follows the nonlinear approximation paradigm by coding only the largest magnitude wavelet coefficients. The model used in the EZW algorithm is the zerotree model. It assumes that the larger wavelet coefficients are more important than the small wavelet coefficients. It is true that the visually significant wavelet coefficients tend to be those coefficients that are largest in magnitude. The zerotree model also assumes that a majority of the coefficients are small. This is also a true assumption for most images. Finally, a tree relationship is assumed between coefficients at different scales. The zerotree algorithm uses the parent child relationship as illustrated in Figure 2.15. If a parent is zero, then it is likely that all of its children are zero. This parent child relationship is referred to as the zerotree paradigm. The zerotree paradigm is a good model as one of the properties of the wavelet transform is persistence across scale.

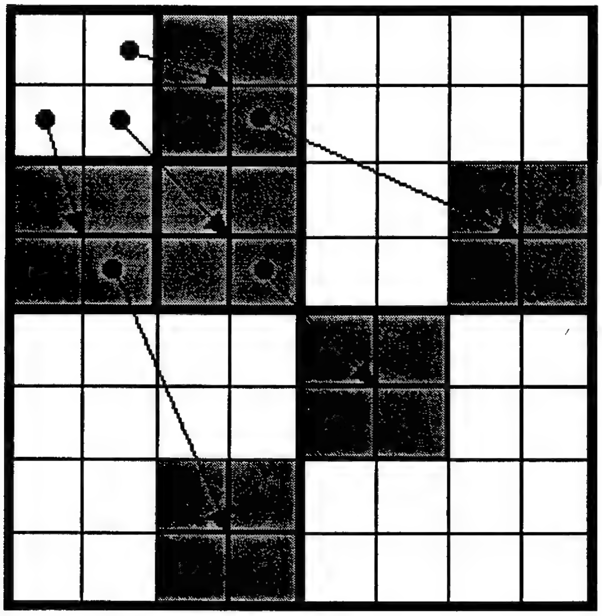


Figure 2.15. *Illustration of the parent child relationship exploited by the EZW.*

An outline of the EZW algorithm is presented below with details in the following paragraphs.

1. Compression:

   (a) Subtract the mean of the image from all pixels within the image to limit the dynamic range.

   (b) Perform the discrete wavelet transform on the zero-meaned image.

   (c) Perform a *dominant pass* (to create a symbol stream.)

   (d) Perform a *subordinate pass* (to create the bit-plane stream.)

   (e) GoTo Step 1c until a target bit rate is achieved, a given PSNR has been reached, or until a prescribed number of iterations has been performed.

   (f) Entropy code the symbol stream.

   (g) Create a header to include the image mean, initial threshold value, image size, and the iteration level of the DWT.

(h) Concatenate the symbol stream, bit planed bit stream, and the header to create a single stream of zeros and ones that is now the new representation of the current image.

2. Reconstruction:

   (a) Extract the symbol stream, bit planed bit stream, and the header information from the stream of zeros and ones that is the image representation.

   (b) Decode the image mean and the initial threshold value $T$.

   (c) Decode the entropy coded symbol stream.

   (d) Perform an *inverse dominant pass* to reconstruct the image structure and position of the significant wavelet coefficients.

   (e) Perform an *inverse subordinate pass* to reconstruct the values of the significant coefficients.

   (f) GoTo Step 2d until there is no more information available to process.

   (g) Perform the inverse discrete wavelet transform.

   (h) Add the mean of the image back to each pixel within the image.

*2.9.1.1 EZW Compression.* The compression portion of the EZW algorithm contains three major pieces: *Dominate Pass*, *Subordinate Pass*, and coding the resulting data streams. The bulk of the work is accomplished in these subcomponents of the EZW algorithm.

*Dominant Pass.* The dominate pass finds the largest magnitude wavelet coefficient and sets the initial threshold value $T$ to the largest power of two less than the largest wavelet coefficient. The threshold value can be determined mathematically using Equation 2.23.

$$T = 2^{\lfloor \log_2(\max(|w_i|)) \rfloor} \tag{2.23}$$

where $\lfloor x \rfloor$ is defined as the floor function. The floor function returns the largest integer value smaller than or equal to $x$. After $T$ has been calculated, it is compared with each wavelet coefficient $w_i$. All $|w_i|$ strictly less than $T$ are set to zero. All $w_i$ whose magnitudes are greater than or equal to $T$ are considered "significant." If the value of the significant is greater than zero, then the significant coefficient is *positive significant.* If the value is less than zero, then the coefficient is *negative significant.* The scanning order is important in this algorithm. All coefficients of a particular scale are checked before the coefficients of the next scale. The scanning order is presented in Figure 2.16. If an insignificant value is found, than it is considered a zero and its children must then be checked. If the children at all scales are zero, then it is a parent of a zerotree root. If $| w_i |$ is less than $T$ and its children at all scales are not zero, then the coefficient is considered an isolated zero. Two vectors are maintained throughout the dominant pass. One vector maintains the magnitudes of the significants while the other vector maintains the symbols: PS, NS, IZ, and ZTR (for *positive significant*, *negative significant*, *isolated zero*, and *zerotree root* respectively.)

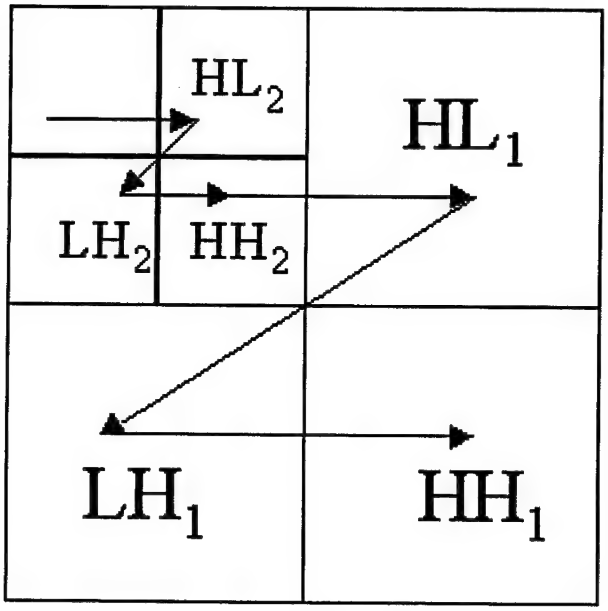


Figure 2.16. *Scanning order of wavelet coefficients with 2 scales.*

After the entire image has been scanned in the manner discussed above, the vector of coefficient magnitudes is sent to the subordinate pass in order to generate the bit plane coding of the significant values. The symbol stream vector (which contains the PS, NS, IZ, and ZTR symbols) is appended to the symbol stream vector from previous iterations of the Dominant Pass. After the subordinate pass has finished, the threshold value is halved and the process described in this section is repeated until either a target bit rate, desired PSNR, or target number of iterations has been achieved.

*Subordinate Pass.* The subordinate pass uses the magnitude information found during the dominant pass. Its sole function is to bit plane code the significant values: initial bit assignment and bit refinement. See Section 2.9 for information on bit plane coding.

#### *2.9.1.2 Reconstruction.*

*Inverse Dominant Pass.* The inverse dominant pass traverses the image as described in Section 2.9 (see Figure 2.16) using the symbol vector to place reconstructed significant values and zeros values due to isolated zeros (IZ) or zerotree roots (ZTR.) The reconstruction values of the significants are determined by the inverse subordinate pass. If the value is significant, the dominant pass uses the symbol to determine if the reconstruction value is positive or negative. If the value is significant and the symbol is a PS, then that reconstruction value is positive. If the value is significant and the symbol is an NS, then that reconstruction value is negative.

*Inverse Subordinate Pass.* As with the subordinate pass, which was responsible for bit plane coding the significant coefficients, the inverse subordinate pass reconstructs the significant coefficients. In order for this to occur, the subordinate pass needs the bit plane stream of bits along with the maximum threshold $T_{max}$ and the number of significant coefficients per pass (which are known due to the use of the symbol stream and the specific scanning order of the EZW algorithm.)

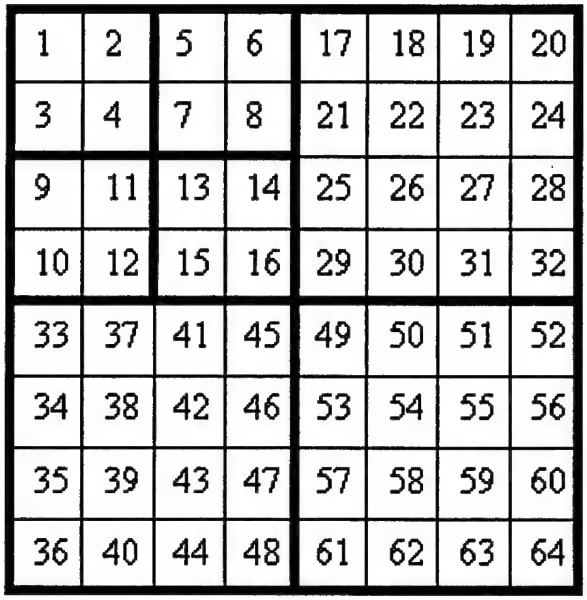
*2.9.2 Image Compression Using Index Coding.* Tian et al (23) have developed a compression method which finds significant wavelet coefficients of the discrete wavelet transformed image and codes their location index. This method uses some of the same concepts as the zerotree method: locating coefficients larger than some threshold $T$ and bit plane coding the significant coefficients as described in Section 2.9. The model used in this compression scheme follows the parsimony property of the DWT; the largest wavelet coefficients contain a majority of the visually significant information within the wavelet transformed image and that there are small number of these coefficients. Furthermore, significant coefficients in the same scale are generally close together. The main advantage of this method is that it is more efficient than the EZW method presented above, and slightly more effective (according to the results presented in (23).) Before explaining the algorithm in more detail, it is necessary to discuss index coding.

Index coding in this compression algorithm uses the first difference between adjacent values. In order for the first difference technique to work, the index values must be monotonically increasing, thus alleviating the need to code negative numbers. Consider the set of index values $S = \{1\ 4\ 7\ 19\ 22\}$. The first difference between adjacent values is performed to create the set $S' = \{1\ 3\ 3\ 12\ 3\}$. To reconstruct the indices of $S$ using $S'$, all previous values of are summed (see Equation 2.24)

$$i(k) = \sum_{l=1} k(l). \tag{2.24}$$

The compression technique works on the discrete wavelet transform of the zero-meaned image. The initial threshold $T_{init}$ is set to the largest power of two smaller than the maximum wavelet coefficient (as see Equation 2.23.) The image is scanned in the order presented in Figure 2.17 to minimize the distances between the indices of significant coefficients. If a wavelet coefficient is found that is greater than or equal to $T_{current}$, then it is significant and is recorded along with its position within the image. This process is repeated until a target bit rate or a specific PSNR is reached.

The magnitudes of the significant coefficients are bit plane coded discussed in Section 2.9 and the indices of the significant coefficients are coded using the first-difference. After the first difference is calculated, the signs of the significants are interleaved with the indices. For example, say the significant coefficients are $C = \{42\ -33\ 19\,12\ -8\}$ and the locations of these significant are $S = \{1\ 4\ 7\ 19\ 22\}$. This means that $S' = \{1\ 3\ 3\ 12\ 3\}$ and $S'$ interleaved with the sign information is $P = \{+1 - 3 + 3 + 12 - 3\}$.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 5 | 6 | 17 | 18 | 19 | 20 |
| 3 | 4 | 7 | 8 | 21 | 22 | 23 | 24 |
| 9 | 11 | 13 | 14 | 25 | 26 | 27 | 28 |
| 10 | 12 | 15 | 16 | 29 | 30 | 31 | 32 |
| 33 | 37 | 41 | 45 | 49 | 50 | 51 | 52 |
| 34 | 38 | 42 | 46 | 53 | 54 | 55 | 56 |
| 35 | 39 | 43 | 47 | 57 | 58 | 59 | 60 |
| 36 | 40 | 44 | 48 | 61 | 62 | 63 | 64 |

Figure 2.17. *Scanning order of wavelet coefficients with 2 scales for index coding based compression.*

The values of the first difference can be converted to binary which yields $P = \{+0001 - 0011 + 0011 + 1100 - 0011\}$. Removing the leading zeros leaves $P = \{+1 - 11 + 11 + 1100 - 11\}$. Every index has a leading 1 (by design) and therefore does not need to be coded. This *binary reduction* on $P$ reduces it to $P = \{+ - 1 + 1 + 100 - 1\}$, significantly less information to code, yet it contains the same information as before.

### *2.10 Entropy Coding*

The entropy $H$, is the average amount of information a particular sequence of symbols contains and is defined as:

$$H = -\sum_{i=1}^{L} P_i \log_2 P_i \qquad (2.25)$$

where $P_i$ is the probability that symbol $a_i$ occurs (12). Additionally, it can be shown that the entropy is bounded by 0 and $\log_2 L$. That is,

$$0 \le H \le \log_2 L. \qquad (2.26)$$

The entropy coding method discussed here is the Huffman coding method. Huffman codes produce a variable length code word based on the probability a given "symbol" occurs. A "symbol" can be an ASCII character set or any unique bit sequence. Huffman coding is desirable because it is easy to use, uniquely decodable, and results in a near optimal bit rate (12). Near optimal means that Huffman coding asymptotically reaches the optimal bit rate which is defined by the entropy $H$. See Appendix D.1 for a detailed example of the Huffman coding process.

### *2.11 Summary*

This chapter provided background information necessary to understand the *Methodology and Design* segment presented in *Chapter 3.* Peak-signal-to-noise ratio (PSNR) was discussed as a method of measuring image quality. General 1–$D$ wavelet theory was provided followed by a brief discussion of biorthogonal filters and the 2–$D$ wavelet transforms. Models of the human visual system were discussed in order to provide a basic understanding of how watermarking methods take advantage of image properties to hide information. Watermarking characteristics were presented for a basis of digital watermarking techniques. Digital watermarking techniques were covered to bring insight into the watermark based audio insertion strategy used in this thesis. Image compression basics were provided as a precursor to the *Embedded Zerotree Wavelet Algorithm* and *A Lossy Image Codec Based on Index Coding* methods of image compression. Finally, entropy coding was discussed.

# *III. Methodology and Design*

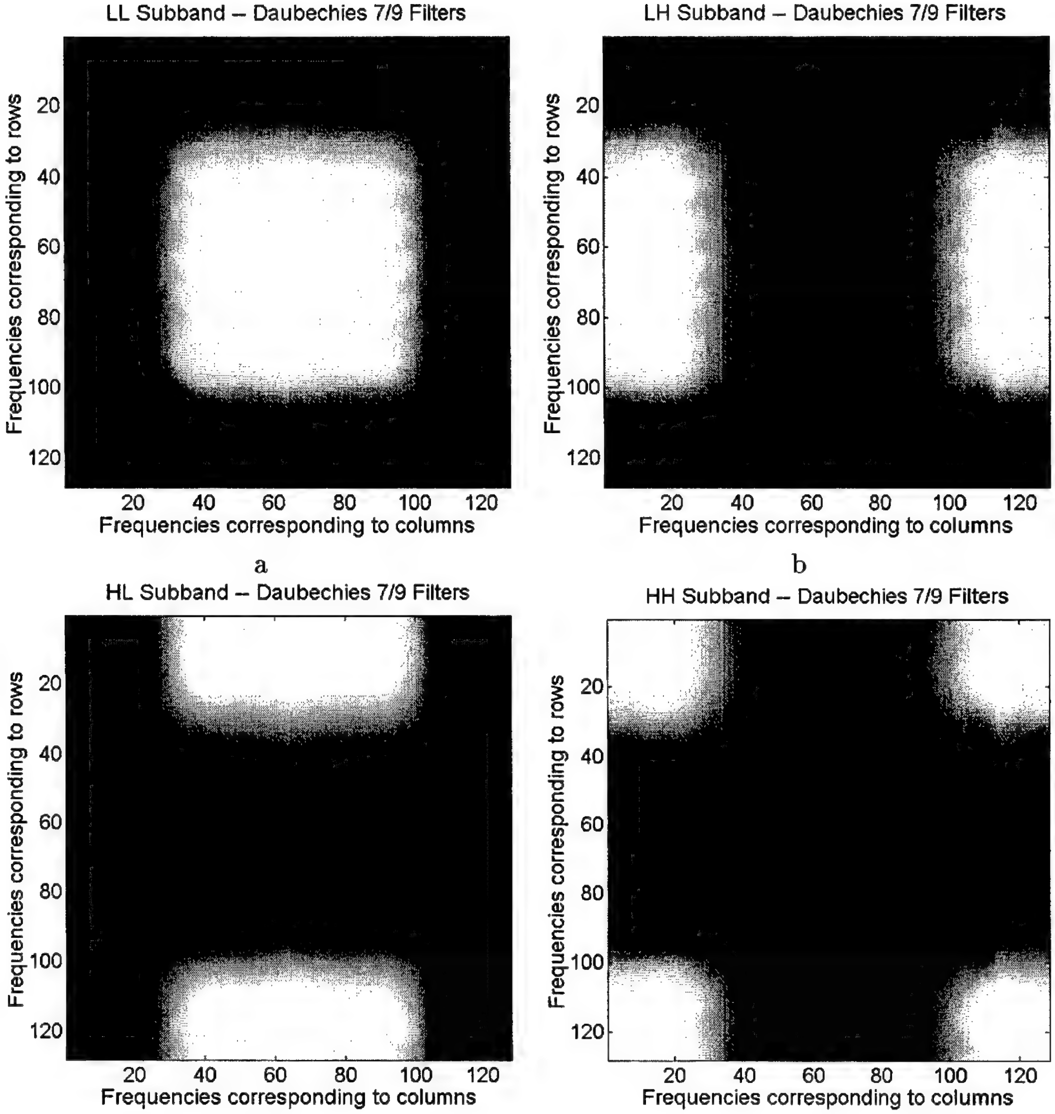
This chapter covers the specific approach to embed audio information into a video sequence. This combined audio/video signal alleviates synchronization problems because these signals are no longer independent. Additionally, the embedding process reduces the size of the audio/video stream because the audio information is incorporated into the individual video frames. By co-designing the audio embedding and audio/video compression technique, further compression gains can be achieved while maintaining the integrity of the audio signal.

## *3.1 Filter Selection*

Several filters are available for use in the discrete wavelet transform (DWT). When choosing filters, frequency response is an important consideration. The frequency response of Daubechies $(7, 9)$ biorthogonal filters are presented in Figures 3.1a, 3.1b, 3.1c, and 3.1d. The areas in black represent those frequency components that are filtered, those regions in white represent those frequency components that are passed. The region between the black and white areas is called the transition band. The transition band for the Daubechies $(7, 9)$ biorthogonal filters is small. Small transition bands are desirable because they reduce the amount of undesirable frequency content.

Four filter coefficient sets are considered: Haar orthogonal filters, Daubechies 4 orthogonal filters, Daubechies 8 orthogonal filters, and Daubechies $(7, 9)$ biorthogonal filters. Figure 3.2 demonstrates that the Daubechies $(7, 9)$ filters perform well when reconstructing an image by keeping only significant coefficients (those coefficients with the largest magnitude.)

It is known that Daubechies $(7, 9)$ filters behave very well. They are linear phase filters that have good time/frequency localization and the resulting wavelet coefficients decay rapidly (a result of the frequency response.) Daubechies $(7, 9)$ biorthogonal filters are near orthogonal. Orthogonal filters leave noise uncorrelated in the wavelet domain. Hence, the noise is white which is indicated by a diagonal covariance matrix. Biorthogonal filters tend to correlate noise in the wavelet domain. Hence, noise is colored, indicated by



a

b

c

d

Figure 3.1. *Frequency response for the Daubechies* (7, 9) *biorthogonal filters. a) LL subband has high-frequency components filtered. b) LH subband preserves vertical edges. c) HL subband preserves horizontal edges. d) HH subband preserves* 45° *edges.*

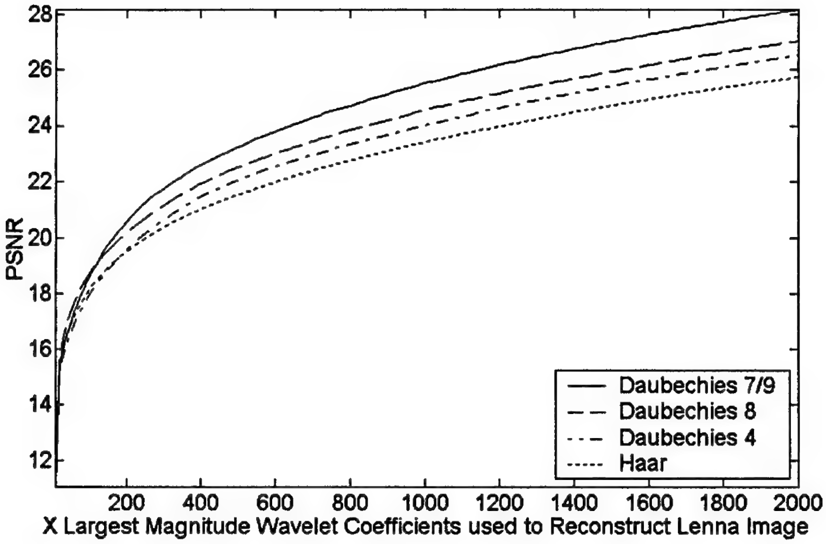


Figure 3.2. *PSNR-vs-Number of Coefficients. Here the number of coefficients refers to the N largest magnitude coefficients used to reconstruct the* 256 × 256 *pixel "Lenna" image and range from* 10-2000 *in steps of* 10 *coefficients. The PSNR plot has four curves. From top to bottom, these curves represent the reconstruction of the "Lenna" image using: Daubechies* (7, 9) *biorthogonal filters, Daubechies* 8 *orthogonal filters, Daubechies* 4 *orthogonal filters, and the Haar orthogonal filters.*

off diagonal terms in the covariance matrix. Daubechies (7, 9) biorthogonal filters have the advantages of biorthogonal wavelets, yet they leave the noise mostly uncorrelated. Hence, the noise is slightly colored indicated by near zero off diagonal terms. Additionally, it is generally accepted in the image processing community that Daubechies (7, 9) are the preferred filters for image processing functions such as image compression. The Daubechies (7, 9) biorthogonal filters are presented in Figure 3.3. For these reasons, Daubechies (7, 9) biorthogonal filters are used in this thesis.

### *3.2 Watermarking*

There are four methods for audio embedding proposed in this research. These methods are based on the discrete wavelet transform of the receiving image. The first two methods of audio embedding use techniques in digital watermarking. The watermark in this instance is not an image, but rather a binary audio stream. The third method of audio embedding is an encoding method based on the zerotree wavelet algorithm devel-

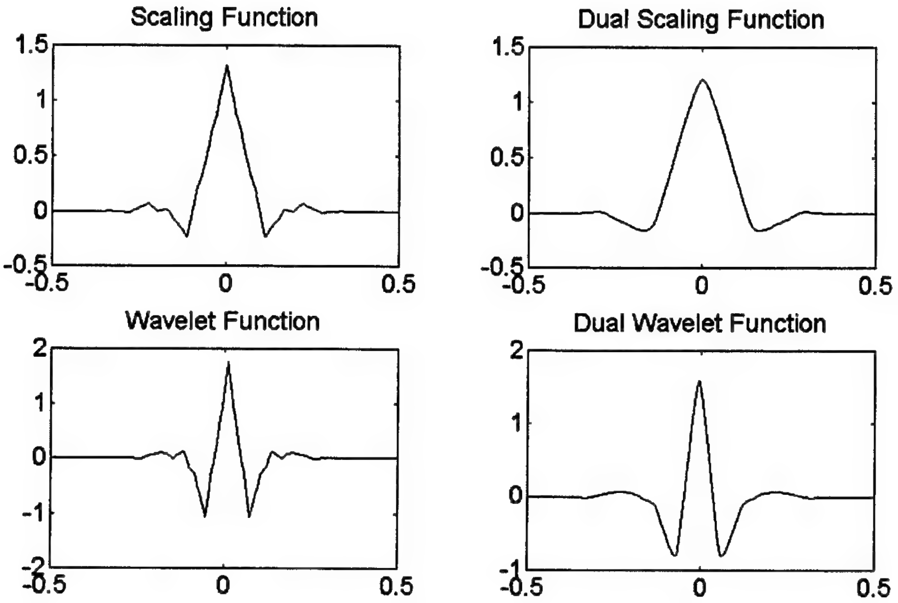


Figure 3.3. *Daubechies* (7, 9) *biorthogonal filters. The scaling and wavelet functions correspond to the synthesis bank filters. The dual wavelet and dual scaling functions correspond to the reconstruction bank filters. The smoothness of the dual scaling and dual wavelet functions for the reconstruction bank is a significant component of the superior performance of the Daubechies* (7, 9) *biorthogonal filters.*

oped by Jerome M. Shapiro (18). It is important to note that watermarking methods are true embedding methods as they physically alter the affected wavelet coefficients. The zerotree method is an encoding method as it encodes the audio as it compresses the given image. Finally, the fourth method is a hybrid method that incorporates watermarking and compression.

As discussed in Section 2.8.1, there are certain characteristics a watermarking technique should possess. The primary characteristics are *imperceptibility* and *robustness.* These primary characteristics conflict and a tradeoff between the two must be reached. In addition to the primary characteristics there are secondary characteristics. The two important secondary characteristics for the audio embedding problem are: *extraction without original information* and *real-time processing.* For these reasons, the modular-based thresholding embedding and extracting algorithm developed by Tsai et al (24) is used. Although the implementation of the embedding algorithms below are not real-time, they lend themselves to real-time processing because of their simplicity and non-mathematically intensive nature (aside from performing the discrete wavelet transform.) The watermarking technique described below requires very little additional information to extract the watermark limiting the amount of overhead involved in storing or transmitting the combined audio/video information (a must for image compression.) The modular-based thresholding algorithm used to embed and extract the watermark information developed by Tsai et al is presented below.

1. Embedding Algorithm (24):

    (a) if $A == 0$

        i. if $W \geq 0$
        then $W' = W - |W| \bmod S + T_2$

        ii. else $W < 0$
        then $W' = W + |W| \bmod S - T_2$

    (b) else $A == 1$

        i. if $W \geq 0$
        then $W' = W - |W| \bmod S + T_1$

ii. else $W < 0$

then $W' = W + |W| \bmod S - T_1$

2. Extraction Algorithm (24):

   (a) if $|W'| \bmod S \geq \frac{T_1+T_2}{2}$

   then $A = 0$

   (b) else $|W'| \bmod S < \frac{T_1+T_2}{2}$

   then $A = 1$

3. Data Dictionary:

   (a) $W$ is the current wavelet coefficient that is going to receive the watermark.

   (b) $W'$ is the wavelet coefficient modified with the watermark data information.

   (c) $A$ is the watermark bit.

   (d) $S$ is a modular value.

   (e) $T_1$ and $T_2$ are threshold values (usually a function of $S$) that govern the embedding of the 1s and 0s respectively.

Two methods of watermarking are explored using the embedding and extracting technique described above. The first is a two-dimensional audio embedding approach while the second is a one-dimensional audio embedding approach. These two approaches are described in detail below.

*3.2.1 2–D Embedding.* The 2–$D$ audio embedding technique is modeled after the methods described in Tsai et al's (24) work. The work accomplished by Tsai et al is a joint spatial/wavelet domain version of a purely spatial watermarking method developed by Voyatzis et al (26). In the algorithm presented below, the audio information (treated as a watermark) is inserted into a given subband of a given scale. Figure 3.4 illustrates that the visually significant regions of the image are the bandpass subbands. The "checkerboard" pattern in quadrant 1 represents a binary audio image. Further discussion of this algorithm is presented in Section 3.2.5. The 2–$D$ watermarking method is presented below:

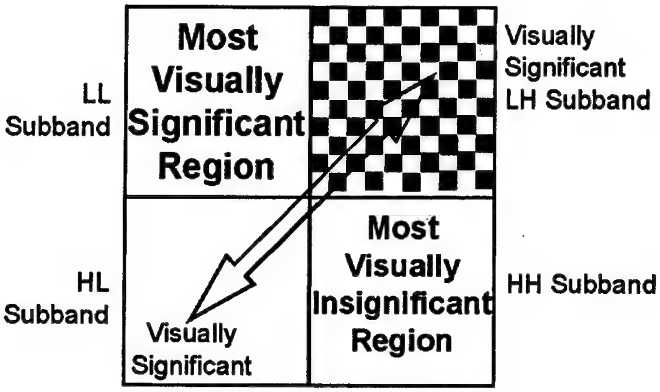

Most Visually Significant Region
LL Subband
Visually Significant LH Subband
HL Subband
Visually Significant
Most Visually Insignificant Region
HH Subband

Figure 3.4. *"Visually significant" regions are the bandpass subbands. In the dyadic decomposition of the wavelet transform, four subbands are generated: the LL subband which corresponds to the coarse approximation, the HL and LH subbands referred to as the bandpass subbands, and the HH subband which corresponds to the detail subband. When processing in the wavelet domain, the coarse approximation contains a likeness of the original image. Modifying these coefficients generally has a negative effect on image quality. For these reasons, the LL subband is considered the "most visually significant region" of the image. The bandpass subbands (HL and LH subbands) contain a mixture of information (varying levels of detail within the image) and are therefore considered "visually significant" regions within the image.*

1. Description: The algorithm below defines a method to embed the audio information into various wavelet coefficients. This algorithm is iterative.

2. Data Dictionary:

   (a) $I$ is the input image.

   (b) $I^{mean}$ is the mean of the image.

   (c) $I^{temp}$ is a temporary variable used to store a modified version of $I$.

   (d) $W$: $W = \mathcal{DWT}(I^{temp})$. Note: $\mathcal{DWT}(I^{temp}, L)$ indicates that $L$ iterations of the $\mathcal{DWT}$ should be performed.

   (e) $A$ is the binary audio data embedded as a watermark within $I$.

   (f) $W^{mod}$ is the set of wavelet coefficients modified with the data in $A$.

   (g) $E$: $E \subset W$. $E$ is the set of wavelet coefficients that receive the watermark information in $A$.

(h) $E^{mod}$: $E^{mod} = \mathcal{F}(E, A)$. Here, $\mathcal{F}(E, A)$ is the embedding process defined Section 3.2.

(i) $S$ is an integer value used for a modulus operation. As $S$ increases, the imperceptibility of the watermark decreases, but the robustness of the watermark increases. As $S$ decreases, the imperceptibility of the watermark increases, but the robustness of the watermark decreases.

(j) $(p_1, p_2)$ is the insertion point of the $N \times N$ audio "image."

(k) $N$ is the size of the audio image ($N \times N$.)

(l) $T_1$ is a threshold value used to embed the watermark into $I$.

(m) $T_2$ is a threshold value used to embed the watermark into $I$.

(n) $a$ is an index counter for the outer for loop.

(o) $b$ is an index counter for the inner for loop.

(p) $index_{audio}$ is an index to the current location within the array of audio coefficients.

(q) $index_{video}$ is an index to the current location within the array of video frames.

3. Functional Description:

(a) $index_{audio} = 1$

(b) $index_{video} = 1$

(c) $S = 32$: Work shown by Tsai et al indicates that this is a good value for $S$ (24).

(d) $T_1 = \frac{3*S}{4} = 30$: Work shown by Tsai et al indicates that this is a good value for $T_1$ (24).

(e) $T_2 = \frac{S}{4} = 10$: Work shown by Tsai et al indicates that this is a good value for $T_2$ (24).

(f) *Init*:

```
i. if $index_{video} > length(I)$
   then return
   else continue
```

ii. $I^{mean} = mean(I(index_{video}))$: Get the mean of the current frame.

iii. $I^{temp} = I(index_{video}) - I^{mean}$: Zero mean the current frame.

iv. $W = \mathcal{DWT}(I^{temp}, L)$: Choose $L = 6$ to perform 6 iterations of the $\mathcal{DWT}$.

v. $E$ equals an $N \times N$ sub-matrix of $W$ starting at some point $(p_1, p_2)$ i.e., $E = W(p_1 : p_1 + N, p_2 : p_2 + N)$: $E$ is typically a given subband of a particular band within $W$. For example, the LH subband of band 1 of a a 256 × 256 pixel image would be a 128 × 128 pixel area consisting of $128^2 = 16,384$ elements.

(g) *Embed*:

i. for $a = 1$ to $N$

ii. for $b = b$ to $N$

iii. if $index_{audio} > length(A)$

then GoTo *Post Processing*

else continue

iv. if $A_{(a-1)*N+b} == 0$

- A. if $E_{a,b} \geq 0$

  then $E^{mod}_{a,b} = E_{a,b} - |E_{a,b}| \bmod S + T_2$

- B. else $E_{a,b} < 0$

  then $E^{mod}_{a,b} = E_{a,b} + |E_{a,b}| \bmod S - T_2$

v. else $A_{(a-1)*N+b} == 1$

- A. if $E_{a,b} \geq 0$

  then $E^{mod}_{a,b} = E_{a,b} - |E_{a,b}| \bmod S + T_1$

- B. else $E_{a,b} < 0$

  then $E^{mod}_{a,b} = E_{a,b} + |E_{a,b}| \bmod S - T_1$

vi. $index_{audio} = index_{audio} + 1$

vii. $index_{video} = index_{video} + 1$

(h) *Post Processing*:

i. $W^{mod} = W - E + E^{mod}$: Replace the old values of $E$ within $W$ with $E^{mod}$.

ii. $I^{mod}(index_{video}) = \mathcal{DWT}^{-1}(W^{mod}, L)$: Perform the inverse discrete wavelet transform on $W^{mod}$.

iii. $I^{mod}(index_{video}) = I^{mod}(index_{video}) + I^{mean}$: Add the mean back into the image.

iv. $I^{mod}(index_{video}) = quantize(I^{mod}(index_{video}))$: Quantize the pixels of the modified frame.

v. $index_{video} = index_{video} + 1$: Increment the video frame array index.

vi. GoTo *Init*

*3.2.2 2–D Extraction.* Very little additional information is needed in order to extract the audio information using this 2–$D$ extraction approach. In order to extract the watermark, the modular value $S$, thresholding parameters $T_1$ and $T_2$, and the insertion location $(p_1, p_2)$ are required. Although the dimensions of the watermark could be helpful, as long as the number of iterations of the DWT is known, then based on the insertion point $(p_1, p_2)$, the size of the watermark can be determined.

The process for audio extraction is to perform $L$ iterations of the DWT on each of the video frames and extract the $N \times N$ region starting at point $(p_1, p_2)$. Using the values of $S$, $T_1$, and $T_2$, the watermark is extracted using the procedure outlined in Section 3.2. The 2–$D$ extraction algorithm is presented in detail in Appendix F.1.

*3.2.3 1–D Embedding.* In this approach, the binary audio data is embedded into the $N$ largest magnitude wavelet coefficients (ignoring those coefficients that make up the coarse approximation.) These $N$ largest magnitude wavelet coefficients represent the most visually significant information within a given video frame.

The audio embedding process is similar to that performed in the 2–$D$ audio approach. First, $L$ iterations of the wavelet transform are performed on the current video frame. Next, the $N$ largest magnitude wavelet coefficients are located (ignoring the coarse approximation.) This set of $N$ coefficients are then modified using the embedding technique described in Section 3.2. Appendix F.2 describes the 1–$D$ audio embedding algorithm in detail.

*3.2.4 1–D Extraction.* As with the 2–$D$ audio extraction approach, the 1–$D$ audio extraction approach uses the modular value $S$ and the thresholding parameters $T_1$ and $T_2$. Additionally, this 1–$D$ extraction approach requires the location of the modified wavelet coefficients (those coefficients that were modified from the embedding process.) Simply finding the largest $N$ magnitude wavelet coefficients does not suffice. There is no guarantee that these $N$ coefficients are the correct $N$ coefficients because the embedding process has modified them. Once the proper coefficients have been located, the audio information is extracted using the technique described in Section 3.2. Appendix F.3 describes the 1–$D$ extraction algorithm in detail.

### *3.2.5 Discussion.*

*3.2.5.1 Processing Justification.* The algorithms presented above follow standard processing techniques.

1. The mean is subtracted from the image in order to limit its dynamic range, thus limiting the magnitudes of the wavelet coefficients after the wavelet transform is performed on the given video frame.
2. The coarse approximation in the 1–$D$ embedding process is set to zero before selecting the $N$ largest magnitude wavelet coefficients. In doing so, the coarse approximation does not undergo modification during the embedding process. Recall that the coarse approximation is the set of wavelet coefficients that have undergone lowpass filtering only. They are those coefficients in the upper left region of the wavelet transformed frame (see Figure 3.5.) Additionally, these coefficients represent coarse information within the image and are not spatially located. Modifying these coefficients *could* severely degrade the image quality.
3. The reconstructed watermarked image is quantized because the pixels must be integer values between $0 - 255$ (i.e., 8-bit.)

*3.2.5.2 Satisfying Watermarking Characteristics.* The primary characteristics, *imperceptibility* and *robustness*, are incorporated in the modular value $S$. As discussed

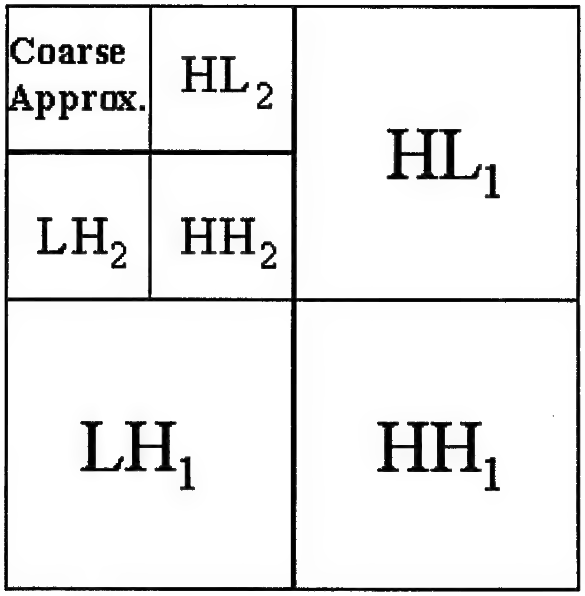


Figure 3.5. *The upper left hand corner represents the coarse approximation to the image.*

in Section 2.8.1, the two primary characteristics conflict and it is necessary to achieve a balance between the two. A small value of $S$ corresponds to a high level of *imperceptibility* and a low level of *robustness.* A large value of $S$ corresponds to a low level of *imperceptibility* and a high level of *robustness.* As indicated by Tsai et al (24), and experimentally verified, a value of $S$ around 32 seams to have a "good" balance of *imperceptibility* and *robustness.*

The two secondary characteristics that are important in this research are *extraction without original information* and *real-time processing.* As indicated in the algorithm descriptions above, if $S$, $T_1$, $T_2$, and the locations of the embedded coefficients are known, then the extraction without original information characteristic is met. Although the Matlab implementation is not real-time (as Matlab is an interpretive language and tends to be slow), the embedding and extraction techniques are not mathematically intensive and do not require a large amount of processing. If the watermarking techniques were implemented in C, C++, or potentially hardware, then it is is possible that the watermark embedding and extraction techniques could be accomplished in real-time.

*3.2.5.3 Advantages/Disadvantage.* There are two main advantages of the 2–$D$ audio embedding approach over the 1–$D$ audio embedding approach. These are overhead and simplicity. The overhead in the 1–$D$ approach involves maintaining a list of indices to keep track of the locations where the audio information is embedded. The list of indices can be quit long and expensive to store or transmit. The simplicity in the 2–$D$ algorithm is due to embedding the audio information in a region starting at a single point.

Conversely, the main advantage of the 1–$D$ audio embedding approach over the 2–$D$ audio embedding approach is that the audio information is embedded in the wavelet coefficients with the largest magnitude, which represent the most visually significant coefficients (not including the coarse approximation.) The 1–$D$ approach also follows the nonlinear paradigm used in image compression. It therefore provides a better avenue for using pre-existing compression techniques, or creating a new compression technique that takes advantage of the information contained in the $N$ largest magnitude wavelet coefficients.

*3.2.5.4 Migration to Compression Based Methods.* Now that there are several methods to embed and extract the audio information from a video frame, the next step is to compress the combined audio/video signal. The problem is that there is potential for modifying wavelet coefficients due to the compression technique that would cause errors when extracting the audio information. It is necessary to either modify an existing compression technique or create a new compression technique that ensures the wavelet coefficients embedded with the audio information are present and not modified in such a manner that introduces audio bit errors during extraction. In either case, the 2–$D$ watermarking approach is no longer viable because it is highly unlikely that all of the visually significant information lies within a particular subband of the discrete wavelet transformed video frame. The 2–$D$ audio embedding approach follows the linear paradigm and is not well suited to the wavelet transform.

Many discrete wavelet transform based compression techniques take advantage of the fact that the largest wavelet coefficients contain the visually significant portions of the image. This is the same principle the 1–$D$ audio embedding approach follows. Figure 3.6 illustrates this principle by displaying the "Lenna" image reconstructed using only the

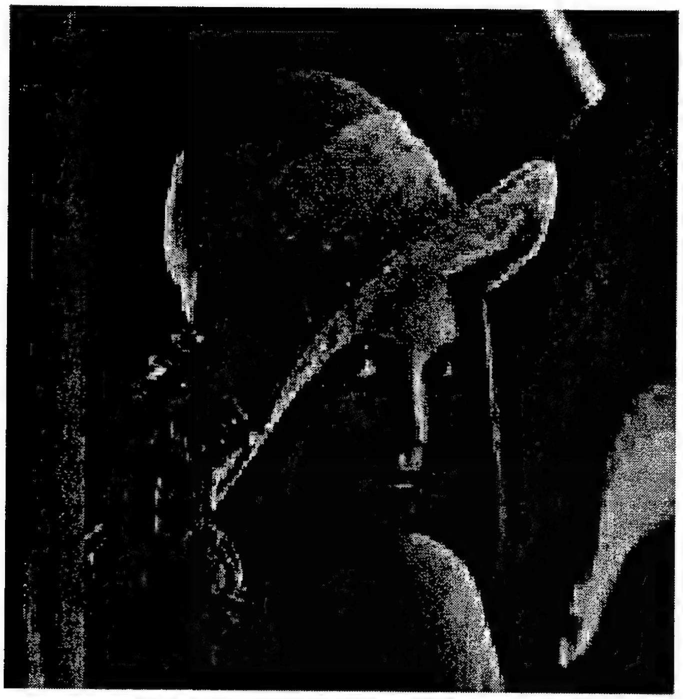


Figure 3.6. *Reconstructed "Lenna" image created by keeping the* 3000 *largest magnitude wavelet coefficients.*

3000 largest magnitude wavelet coefficients. The PSNR of the reconstructed "Lenna" image is 29.96dB. In order to have an effective audio embedding and audio/video compression system, it is necessary to design the system with both objectives in mind. The 1–$D$ audio embedding method searches for the wavelet coefficients with the largest magnitude. As alluded to previously, many wavelet based compression schemes do the same. One such compression technique is the *Embedded Zerotree Wavelet Algorithm* described in Section 2.9.1.

### *3.3 Audio Encoding and the Embedded Zerotree Wavelet Algorithm*

Simply compressing the audio/video signal using the EZW algorithm results in a failure of the extraction technique to correctly extract the audio information. The problem lies in how the EZW determines the reconstruction values of the wavelet coefficients (see Section 2.9.1) using bit plane coding. By modifying a wavelet coefficient before executing the EZW, it might jump threshold boundaries resulting in a different reconstruction level for that coefficient. Because the EZW algorithm bit planes the coefficients, there is no feasible way to embed the audio information after the wavelet coefficients are located for the current threshold. The following section describes how the EZW algorithm is modified in order to encode the audio information as the video data is compressed.

*3.3.1 Modifications to the EZW.* Recall the embedded zerotree wavelet (EZW) algorithm described in Section 2.9.1. There were a total of four symbols in the symbol set: *isolated zero* (IZ), *zerotree root* (ZTR), *positive significant* (PS), and *negative significant* (NS.) Some variations of the EZW algorithm create additional symbols such as *positive significant zerotree root* (PSZTR) and *negative significant zerotree root* (NSZTR) to indicate that the given PS or NS is the parent of a zerotree root. In this example, higher compression ratios are gained because the vector of symbols generated during the dominant pass is **smaller** in length.

The audio information is encoded by modifying the symbol set. This is the same avenue pursued to increase the compression rate by creating the PSZTR and NSZTR symbols above. Instead, what is created are additional symbols that represent the encoding of a "1" or a "0". There are several ways the symbol set can be modified. The method used depends on the amount of audio information that needs to be embedded and the amount of compression desired.

The first method is to encode the audio information in the significant values. Encoding the audio in the significants symbols is more desirable if a higher quality video sequence is desired and only a moderate amount of audio information needs to be encoded in a given frame. This is because there are not very many significant coefficients unless a higher bit rate is achieved. A higher bit rate corresponds to a higher quality video sequence. The change is to remove the *positive significant* (PS) and *negative significant* (NS) symbols from the symbol set and replace them with *positive significant zero* (PS0), *positive significant one* (PS1), *negative significant zero* (NS0) and *negative significant one* (NS1). The new symbol set is: IZ, ZTR, PS0, PS1, NS0, and NS1; an addition of two symbols.

The second method is to encode the audio information in the zerotree roots. Encoding the audio in the ZTR symbols is more desirable at high or low target bit rates if a significant amount of audio information needs to be incorporated. A large number of symbols are zerotree roots, especially early in the compression processes when comparing wavelet coefficients against larger threshold values. The change is to remove the ZTR symbol from the symbol set and add a *zerotree root zero* (ZTR0) and a *zerotree root one* (ZTR1) symbol to the symbol set. The new symbol set is: IZ, ZTR0, ZTR1, PS, and NS; an addition of one symbol.

The third method is to encode the audio information in the isolated zeros. Encoding the audio in the IZ symbols is more desirable at medium and high target bit rates. A large number of the symbols are ZTR symbols at low bit rates. As the bit rates increase, there are fewer zerotree roots and more isolated zeros. The change is to remove the IZ symbol from the symbol set and add an *isolated zero-zero* symbol (IZ0) and an *isolated zero-one* symbol (IZ1). The resulting symbol set is: IZ0, IZ1, ZTR, PS, and NS; an addition of one symbol.

Other variations exist. The audio information can be encoded by creating any number of combinations of new symbols used to reflect a hidden audio bit.

*3.3.2 Audio Encoding.* Although Section 3.3.1 discusses the concept of audio encoding using the EZW algorithm, it does not discuss where or how the encoding actions occur. The audio encoding portion of the EZW algorithm is tied closely with the compression portion of the EZW algorithm covered in detail on Section 2.9.1.1. The audio information is encoded during the dominant pass presented in Section 2.9.1.1. The dominant pass now has the responsibility of looking at the current audio bit in the audio bit array. Using the method of encoding the audio information in the significants, the dominant pass would encode the current significant (whether positive or negative) based on the current audio bit. If the audio bit is a zero, then the dominant pass would insert a PS0 or NS0 symbol in the symbol stream vector if the significant was positive or negative respectively. Conversely, if the audio bit is a one, then the dominant pass would insert a PS1 or NS1 symbol in the symbol stream vector if the significant was positive or negative respectively.

In doing so, how is the subordinate pass affected? Recall from Section 2.9.1.1, that the subordinate pass of the EZW algorithm simply generates a bit stream that is a bit planing of the significant values. Because the subordinate pass has no dealings with the symbol stream, no changes to the subordinate pass are required. However, it is necessary to modify the entropy coding method to work on the new symbol set.

#### *3.3.3 Audio Decoding.*

The decoding of the audio information occurs as the video frame is reconstructed. As the dominant pass is responsible for encoding the audio information, the inverse dominant pass is responsible for decoding the audio information from the symbol stream vector, as well as maintaining the array of binary audio data.

As there was no change with the subordinate pass, there is no change again with the inverse subordinate pass. The inverse subordinate pass is still responsible for reconstructing new significant values and refining previous significant values.

### *3.4 Audio Embedding with Audio/Video Compression*

Although the modified EZW algorithm presented above accomplishes audio encoding and audio/video compression, it tends to be very slow (this is characteristic of the EZW algorithm.) Furthermore, the EZW audio **encoding** and audio/video compression technique does not embed the audio information, which is the goal of this research. The next step is to develop a combined audio **embedding** and audio/compression technique.

Recall that the 1–$D$ watermarking technique modifies the $N$ largest magnitude wavelet coefficients as these coefficients contain the most visually significant information within the wavelet transformed video frame. The compression technique must then use all of these coefficients to ensure all of the audio information can be extracted. Furthermore, the reconstructed wavelet coefficients that contain the watermark information must be reconstructed without significant error such that the thresholding based extraction technique described in Section 3.2 can correctly extract the audio information.

There are two problems to overcome for the audio embedding and audio/video compression system to work. First, the $N$ largest magnitude wavelet coefficients before the audio is embedded are very unlikely to be the same $N$ largest magnitude wavelet coefficients after the audio is embedded. This is because the audio embedding technique physically modifies the value of the wavelet coefficients. The compression technique must keep track of which wavelet coefficients are modified. Second, how should the indices to the modified wavelet coefficients be incorporated into the compression scheme?

The solution to the first problem is to extract the largest $N$ magnitude wavelet coefficients along with their indices prior to embedding the audio data. The $N$ coefficients are then watermarked using the $1-D$ watermarking approach. In this method, the coarse approximation ***is*** modified to include audio data. This is done to limit the amount of overhead.

The solution to the second problem is to code the indices of the watermarked coefficients using the index coding technique described in Section 2.9.2. Recall that the order in which the image is scanned is important for index coding. Scanning the image according to Figure 2.17 minimizes the distance between coefficients thus decrease the amount of information used to code those indices.

Recall that *A Lossy Image Codec Based on Index Coding* scans the image and compares values against a threshold value $T$ (as does the EZW algorithm.) Instead of scanning the entire image, the algorithm only operates on the $N$ coefficients extracted and watermarked with the audio data. On each pass, the set of significants are coded using bit plane coding (see Section 2.9.2) and the indices associated with the significants found on the current pass are index coded using the binary reduction process also described in Section 2.9.2. This method of compression follows the non-linear approximation paradigm described in Section 2.9.

In order to ensure that all coefficients are included during the bit plane coding portion of the algorithm, the minimum number of iterations must be determined. The minimum number of iterations can be determined using the largest power of two smaller than the largest wavelet coefficient and the largest power of two smaller than the smallest wavelet coefficient in the set of $N$ watermarked coefficients. The minimum number of passes is calculated using Equation 3.1:

$$NumberOfPasses = \lfloor \log_2(\max(\text{Coeff}) \rfloor - \lfloor \log_2(\min(\text{Coeff})) \rfloor + 1 \tag{3.1}$$

where $\lfloor X \rfloor$ is the floor function which rounds down to the nearest integer value and Coeff are the set of $N$ watermarked wavelet coefficients.

Equation 3.1 determines the minimum number of passes necessary to include all of the $N$ watermarked wavelet coefficients. However, it might be desirable (or necessary) to perform additional bit plane coding passes for two reasons. First, it might be desirable to increase the resolution of the reconstructed wavelet coefficients in order to have a higher quality reconstructed video sequence. Second, it might be necessary to increase the resolution of the reconstructed wavelet coefficients in order to extract the audio information without audio bit errors. For these reason, Equation 3.1 is modified to include an additional parameter which allows the user to increase the number of bit plane encoding passes performed and is reflected in Equation 3.2:

$$\begin{aligned} NumberOfPasses &= \lfloor \log_2(\max(\text{Coeff}) \rfloor - \lfloor \log_2(\min(\text{Coeff})) \rfloor \\ &\quad +1 + PassesOver. \end{aligned} \tag{3.2}$$

After the final bit plane coding pass is completed, the index coded indices are entropy coded. Recall that the index coding process creates a series of symbols: +'s, -'s, 1's, and 0's. These symbols are entropy coded using Huffman coding. This results in a significant reduction in the size of the index coded indices because these symbols can be represented via variable length code (instead of their 8-bit ASCII format.)

After the watermarking, bit plane coding, and Huffman coding processes are finished, the information for the current video frame is coded in the format displayed in Table 3.1. After each frame in the video sequence has been formatted as described in Table 3.1, it is further compressed with an arithmetic coder. The arithmetic coder used in this research is a Unix-based compression utility called *gzip*. For more information on *gzip*, refer to the Unix man pages.

## *3.5 Summary*

This chapter covered the specific approach to embed audio information into a video sequence. Audio/video compression is achieved by co-designing the embedding compression

| Field | Function | Description |
|---|---|---|
| 1 | Mean | Mean of the current frame. |
| 2 | $\log_2$(MaxT) | Maximum threshold of the current frame. |
| 3 | Number of Passes | Number of passes required (desired) to bit plane code the significant coefficients for the current frame. |
| 4 | Number of Coefficients for Pass 1 | Number of significant coefficients found during the first pass of the current frame. |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $n+3$ | Number of Coefficients for Pass $n$ | Number of significant coefficients found during pass $n$ of the current frame. |
| $n+4$ | Bit Plane Code Length | The length of the bit plane coded significant coefficients of the current frame. |
| $n+5$ | Bit Plane Code bits | The actual bits of the bit plane coded coefficients for the current frame. |
| $n+6$ | Indices Length | The length of the first difference Huffman coded indices for the current frame. |
| $n+7$ | Huffman Coded Indices bits | The actual bits from the Huffman coding of the indices for the current frame. |

Table 3.1. *Compression format for each video frame within the video sequence.*

techniques. The resulting audio embedding audio/video compression system is developed in Section 3.4.

Four audio embedding methods were designed. The first method embeds the audio information in a 2–$D$ region of each video frame. The second method embeds the audio information in the $N$ largest magnitude wavelet coefficients. The third method is a modification of the embedded zerotree wavelet algorithm and *encodes* the audio signal as the video signal is compressed. The fourth method uses digital watermarking and several compression methods to embed and compress the audio and video data.

*Chapter 4* pursues the final audio embedding audio/video compression technique developed in Section 3.4. This audio embedding audio/video compression systems uses a digital watermarking technique to embed the audio signal into the video signal. The watermarking technique modifies wavelet coefficients in each frame of the video signal. These modified wavelet coefficients are used in the compression algorithm. The compression algorithm combines several effective compression techniques. First, bit plane coding is used to code the significant wavelet coefficients. Following the nonlinear approximation paradigm, the significant wavelet coefficients are those coefficients with the largest magnitude. Second, the indices to the significant wavelet coefficients are coded using index coding. This index coding technique scans the image in such a manner as to minimize the distance between index values in order to decrease the amount of information needed to reconstruct them. Finally, Huffman coding (a form of entropy coding) is used to code the symbols from the index coding process.

# *IV. Design of Experiments and Results*

This chapter provides information on the design and outcomes of the experimental runs. Information on obtaining and formatting input test data is provided. Parameter values used for audio embedding and audio/video compression are discussed. The frame format for compression discussed in Section 3.4 is covered in more detail to include information on bits allocated for each field.

Following the *Design of Experiments* section, the experimental results are presented and discussed. Details on minimum, maximum, and median PSNR values for each data run is provided. Compression rates and the average number of bits per pixel for each data run is presented. Finally, audio bit errors due to the extraction process is discussed.

## *4.1 Design of Experiments*

*4.1.1 Gathering Input Test Data.* The sample audio and video data used to test the combined audio embedding/compression technique consists of a five second audio and video sequence. The five second video sequence has a frame rate of 30 frames per second resulting in a total of 150 frames of video data. The video format was modified from its original format such that each video frame is 256 × 256 pixels. Each pixel is quantized to 8-bit grayscale. The associated audio data is also five seconds in length. It consists of a left and a right audio channel. Each audio sample of each of the two audio channels is 8-bits.

Each of the 150 video frames is stored as an image. Each image is loaded into Matlab and stored as a plane in a 3–$D$ array. The dimensions of the 3–$D$ array is $256 \times 256 \times 150$. The audio information is stored as a *.wav* file and loaded into Matlab using the *wavread()* command. Matlab represents the *.wav* file as a 2–$D$ columnar array where the first column contains all of the audio samples for the left channel and the second column contains all of the audio samples for the right channel.

*4.1.2 Keeping Wavelet Coefficients.* To determine how many wavelet coefficients should be used in the embedding and compression process, the total length of the binary audio data is divided by 150 (the number of video frames.) The audio data has $40,040$

samples for each of the two audio channels. Before each audio sample is converted to binary, it has to be re-quantized to 8-bits.

The length of the binary audio information is:

$$\begin{aligned}\text{length of audio} &= 8_{\text{bits/sample}} \times 40,040_{\text{samples/channel}} \times 2_{\text{channels}} \\ &= 640,640 \text{ bits}\end{aligned}$$

In order to incorporate all of the audio information, it is necessary to keep the following number of wavelet coefficients from each video frame:

$$\begin{aligned}\text{number of wavelet coefficients} &= \frac{\text{number of audio bits}}{\text{number of video frames}} \\ &= \frac{640,640}{150} \\ &= 4,270.93\end{aligned}$$

If $4,271$ wavelet coefficients are used, frame 150 would be short 10 audio bits. By embedding $4,270$ audio bits per video frame, there is a total of $4,270 * 150 = 640500$ bits, 140 bits less than the total audio sequence. That is, the reconstructed audio information is short

$$\frac{140_{\text{bits/sample}}}{2_{\text{samples/channel}}} = 70_{\text{bits/channel}}.$$

Because each coefficient is 8-bits, there are two "extra" bits for which there is no use. This means that $70 + 2 = 72$ bits of data per channel is lost (note that 72 is a multiple of eight and the audio format is 8-bits per sample.) Hence, the audio sequence is short $\frac{72}{8} = 9$ audio samples per channel. These 9 samples per channel translate to approximately $67ms$ of sound, not enough time difference for the human ear to discern.

### *4.1.3 Choosing Parameter Values.*

*Audio Embedding and Audio/Video Compression.* Recall that both the audio embedding and the audio/video compression techniques are wavelet domain techniques. Therefore, it is necessary to determine the number of iterations of the wavelet transform to perform on the video frames. In order to take advantage of the multiscale properties, the DWT should be performed on a given video frame as many times as possible. However, there is a point at which the filter lengths are larger than the number of rows or columns in the coarse approximation. When this occurs, edge effects become more apparent and the results become meaningless. The general rule of thumb is to determine the number of iterations that would leave one pixel for the coarse approximation (this single pixel would be the mean of the entire image) and perform two iterations less than this number. For a $256 \times 256$ pixel image, a value of $L = \log_2(256) = 8$ would leave one pixel for the coarse approximation. Therefore, $L = 8 - 2 = 6$ iterations of the DWT should be performed. By choosing $L = 6$, the coarse approximation is $4 \times 4$ wavelet coefficients in size. This choice of $L$ is typical for wavelet-based image processing.

*Audio Embedding.* The parameters that control the outcome of the audio embedding are the modular value $S$ and the threshold parameters $T_1$ and $T_2$. Recall that the modular value controls the robustness and imperceptibility tradeoff of the watermarking technique. Large values of $S$ increase robustness and decrease imperceptibility, small values of $S$ decrease robustness and increase imperceptibility. The embedding of the 1s and 0s is controlled by the thresholding parameters.

The values of $T_1$ and $T_2$ are generally a function of $S$:

1. $T_1 = \frac{3S}{4}$
2. $T_2 = \frac{S}{4}$

Several advantages result by choosing the thresholding parameters as above. First, if $S$ is known, then $T_1$ and $T_2$ are known. Second, there is an equal range for which a 1 or a 0 can occupy within the modulus of the wavelet coefficient. That is, any value less than $\frac{T_1+T_2}{2}$ is a 1 and any value greater than $\frac{T_1+T_2}{2}$ is a 0 (where $\frac{T_1+T_2}{2} = \frac{S}{2}$.) Furthermore, Tsai et al (24) have demonstrated that the values of the thresholding parameter presented above allow the extraction of the watermark while balancing the potential of incorrectly extracting a 1 or a 0.

*Audio/Video Compression.* The value of *PassesOver* (*PO*) is the only required parameter for the audio/video compression technique (aside from the number of iterations to perform the DWT.) $PO$ is used to further refine the values of significant wavelet coefficients during the bit plane coding phase. If the minimum number of passes are desired, then set $PO = 0$. However, two circumstances exist that would require a value of $PO \neq 0$. First, it might be more desirable to have more resolution on the reconstructed wavelet coefficients in order to have a more accurate reproduction of the reconstructed video frame. Second, if the value of $S$ is small, then it might be necessary to have more resolution on the reconstructed wavelet coefficients in order to extract the audio information without error. Therefore, the value of $PO$ is determined by the desires of the user.

*4.1.4 Video Frame Compression Format.* The number of bits required to code each field in Table 4.1 was determined empirically from the data runs. The mean field is coded using 8 bits. This is because the mean could never be larger than the largest pixel value, which is an 8-bit value. The maximum threshold is always a power of two and is generally $1,024$, $2,048$, or $4,096$. However, because frames may vary drastically, four bits are used in order to have a maximum threshold that ranges from $2^0 = 1$ to $2^{15} = 32,768$. The number of passes required to bit plane code the significants can be coded using four bits as the number of passes is generally less than 15. However, it could be necessary to change this to five bits if the number of passes were to exceed 15. Typically, 15 passes corresponds to extremely high bit rates. The number of coefficients found during each pass can be coded using 12 bits. The number of significant coefficients found on a given pass is generally under $2^{12} = 4,096$. However, it could be necessary to increase this value to 13 bits in order to code $8,192$ coefficients. The length of the bit plane coded significants can be coded using 15 bits allowing for $32,768$ bits for the bit plane coded significants. Finally, the length of the Huffman coded indices can be coded using 16 bits allowing up to $65,536$ index bits. The bit lengths per field presented in Table 4.1 were determined experimentally and are sufficient for accurately compressing and reconstructing the test data. These bit

| Field | Function | Number of Bits | Description |
|---|---|---|---|
| 1 | Mean | 8 | Mean of the current frame. |
| 2 | $\log_2(MaxT)$ | 4 | Maximum threshold of the current frame. |
| 3 | Number of Passes | 4 | Number of passes required (desired) to bit plane code the significant coefficients for the current frame. |
| 4 | Number of Coefficients for Pass 1 | 12 | Number of significant coefficients found during the first pass of the current frame. |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $n+3$ | Number of Coefficients for Pass $n$ | 12 | Number of significant coefficients found during pass $n$ of the current frame. |
| $n+4$ | Bit Plane Code Length | 15 | The length of the bit plane coded significant coefficients of the current frame. |
| $n+5$ | Bit Plane Code bits | variable | The actual bits of the bit plane coded coefficients for the current frame. |
| $n+6$ | Indices Length | 16 | The length of the first difference Huffman coded indices for the current frame. |
| $n+7$ | Huffman Coded Indices bits | variable | The actual bits from the Huffman coding of the indices for the current frame. |

Table 4.1. *Compression format for each video frame within the video sequence.*

values cover a large enough range that it is likely that they provide excellent performance over a wide range of audio/video data. However, in some situations, these values may not provide enough room to code all of the pertinent information; a more robust formatting method may need to be developed for these cases.

*4.1.5 Experimental Test Runs.* In Table 4.2, runs one and two are compressed only. That is, there is no audio information embedded into the wavelet coefficients. These runs are used to determine how the audio embedding process effects the quality of the video sequence. Because these runs only compress the video data and do not modify the coefficients with the audio data, their reconstructed wavelet coefficients are quantized versions of the real valued coefficients. Therefore, their reconstruction quality should be better than those runs that modify the wavelet coefficients (runs four through fifteen.) Thus, these runs provide a "bound" on the expected performance.

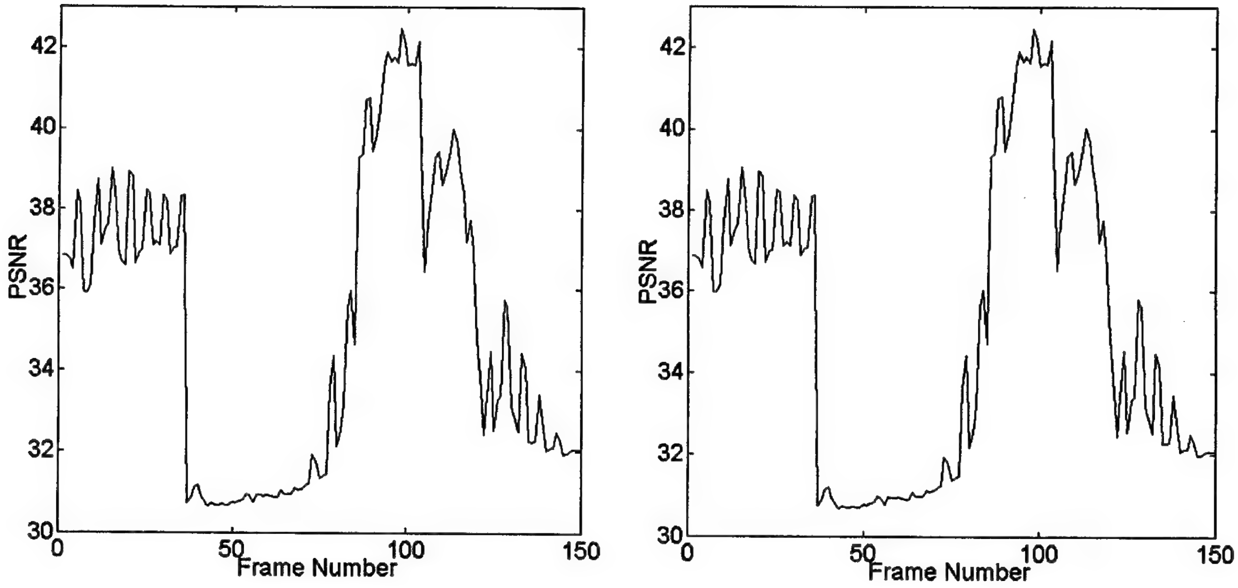
In Table 4.2, runs three through fourteen are accomplished using different parameter values to determine the effects of $S$ and $PassesOver$ on the quality of the reconstructed video sequence, the final compression rate, and the number of audio bit errors. Generally, as $S$ decreases, the PSNR should increase and the compression rate decrease. Additionally, at lower values of $S$, there is more potential for audio bit errors, hence $PO$ is increased in order to increase the resolution on the reconstructed wavelet coefficients.

### *4.2 Results of Experiments*

*4.2.1 Video Results.* The following figures display the PSNR for each of the simulations. The figures are divided into two major categories. The first category, shown in Figures 4.1a and 4.1b, contains those PSNR plots where only video compression is applied. That is, the audio information is not embedded into the video data. This is done to provide comparison data to those compression runs where the wavelet coefficients were modified to include the audio information. The second category, shown in Figures 4.2a, 4.2b, 4.3a, 4.3b, 4.4a, 4.4b, 4.5a, 4.5b, 4.6a, 4.6b, 4.7a, and 4.7b, contains those PSNR plots where the audio data is embedded into the video data and the resulting audio/video signal is compressed. The general shape of the PSNR plots is similar for all data runs.

|  |  | Embedding Parameters—— |  |  |  |
|---|---|---|---|---|---|
| Run No. | $L$ | $S$ | $T_1$ | $T_2$ | $PO$ |
| 1 | 6 | NA | NA | NA | 0 |
| 2 | 6 | NA | NA | NA | 0 |
| 3 | 6 | 40 | 30 | 10 | 0 |
| 4 | 6 | 32 | 24 | 8 | 0 |
| 5 | 6 | 16 | 12 | 4 | 1 |
| 6 | 6 | 16 | 12 | 4 | 0 |
| 7 | 6 | 12 | 9 | 3 | 1 |
| 8 | 6 | 12 | 9 | 3 | 0 |
| 9 | 6 | 8 | 6 | 2 | 1 |
| 10 | 6 | 8 | 6 | 2 | 2 |
| 11 | 6 | 5 | 3.75 | 1.25 | 1 |
| 12 | 6 | 5 | 3.75 | 1.25 | 2 |
| 13 | 6 | 4 | 3 | 1 | 1 |
| 14 | 6 | 4 | 3 | 1 | 2 |

Table 4.2. *Table of parameters for each of the 14 data runs. $L$ is the number of iterations the DWT is performed on the video data. $S$ is the modular value used for the audio embedding and extraction process. $T_1$ and $T_2$ are threshold parameters used in the audio embedding and extraction process. $PO$ is the PassesOver parameter used to perform additional bit plane passes in order to further refine the reconstruction values of the wavelet coefficients.*



a

b

Figure 4.1. *PSNR plots with compression and without audio embedded where: a) PassesOver = 0 and b) PassesOver = 1.*

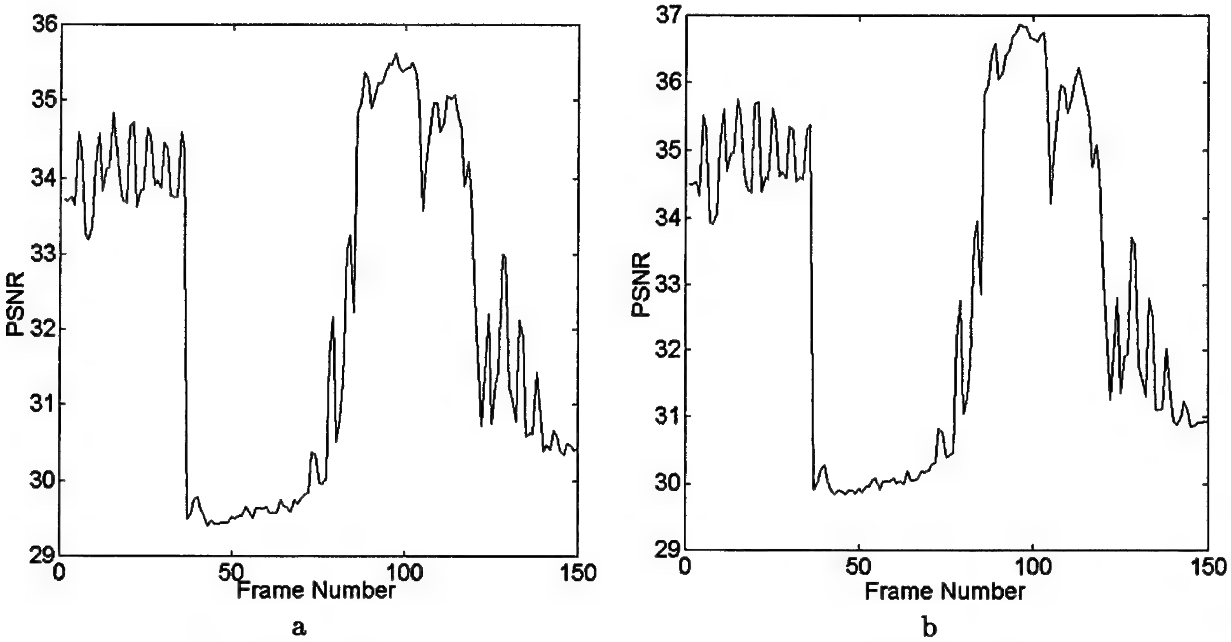


Figure 4.2. *PSNR plots with compression and audio embedding for: a)* $S = 40$, $T_1 = 10$, $T_2 = 30$, *and PassesOver* $= 0$ *and b)* $S = 32$, $T_1 = 8$, $T_2 = 24$ *and PassesOver* $= 0$.

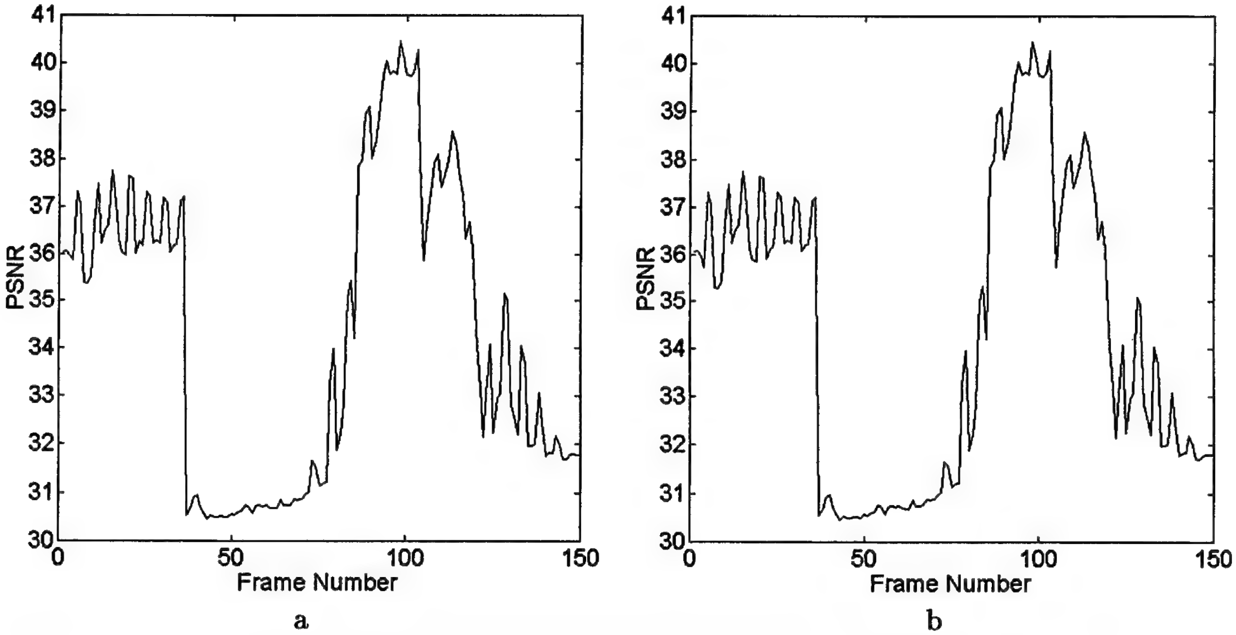


Figure 4.3. *PSNR plots with compression and audio embedding for: a)* $S = 16$, $T_1 = 4$, $T_2 = 12$, *and PassesOver* $= 0$ *and b)* $S = 16$, $T_1 = 4$, $T_2 = 12$, *and PassesOver* $= 1$.

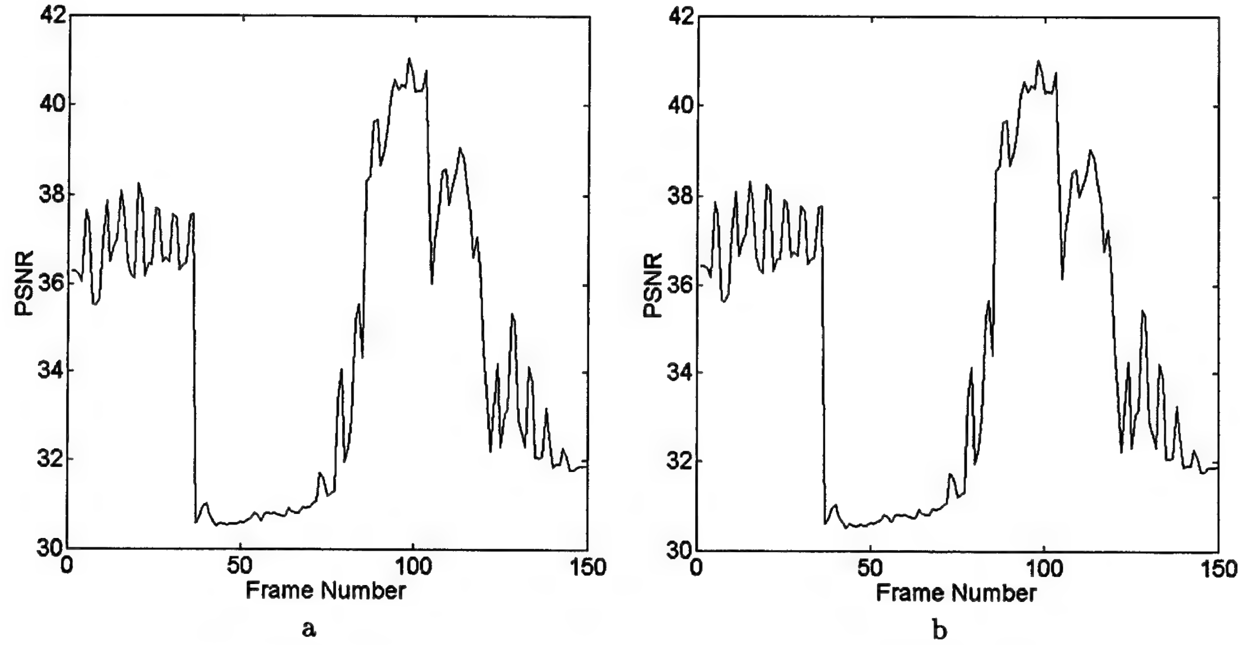


Figure 4.4. *PSNR plots with compression and audio embedding for: a)* $S = 12$, $T_1 = 3$, $T_2 = 9$, *and* $PassesOver = 0$ *and b)* $S = 12$, $T_1 = 3$, $T_2 = 9$, *and* $PassesOver = 1$.

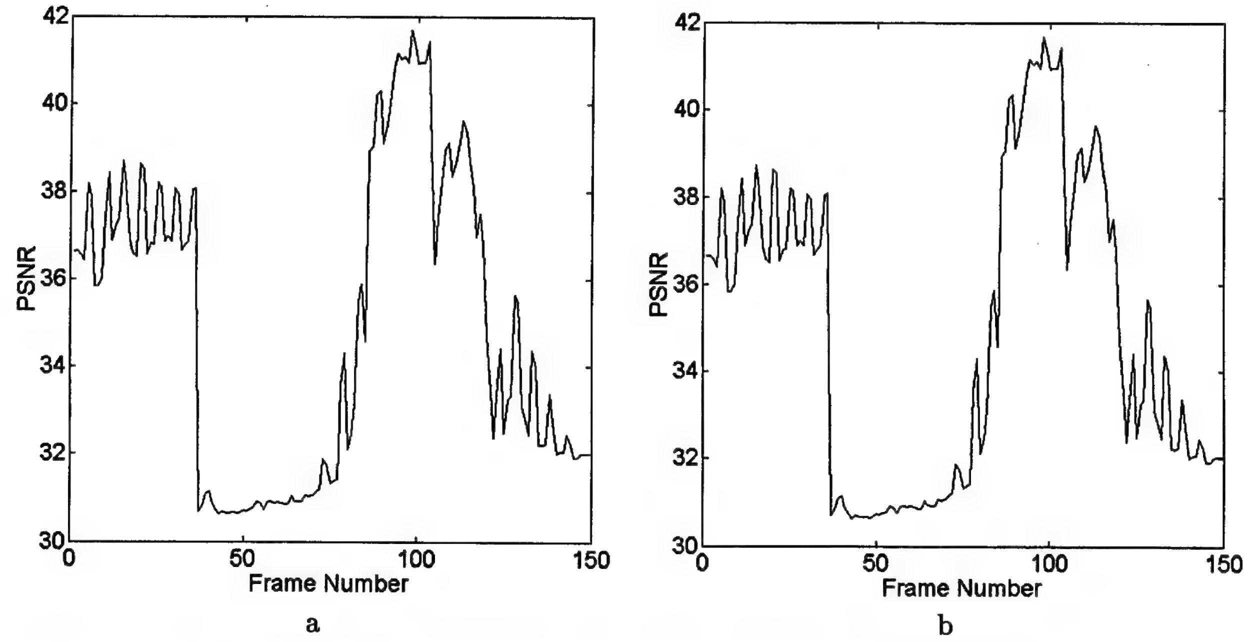


Figure 4.5. *PSNR plots with compression and audio embedding for: a)* $S = 8$, $T_1 = 2$, $T_2 = 6$, *and* $PassesOver = 1$ *and b)* $S = 8$, $T_1 = 2$, $T_2 = 6$, *and* $PassesOver = 2$.

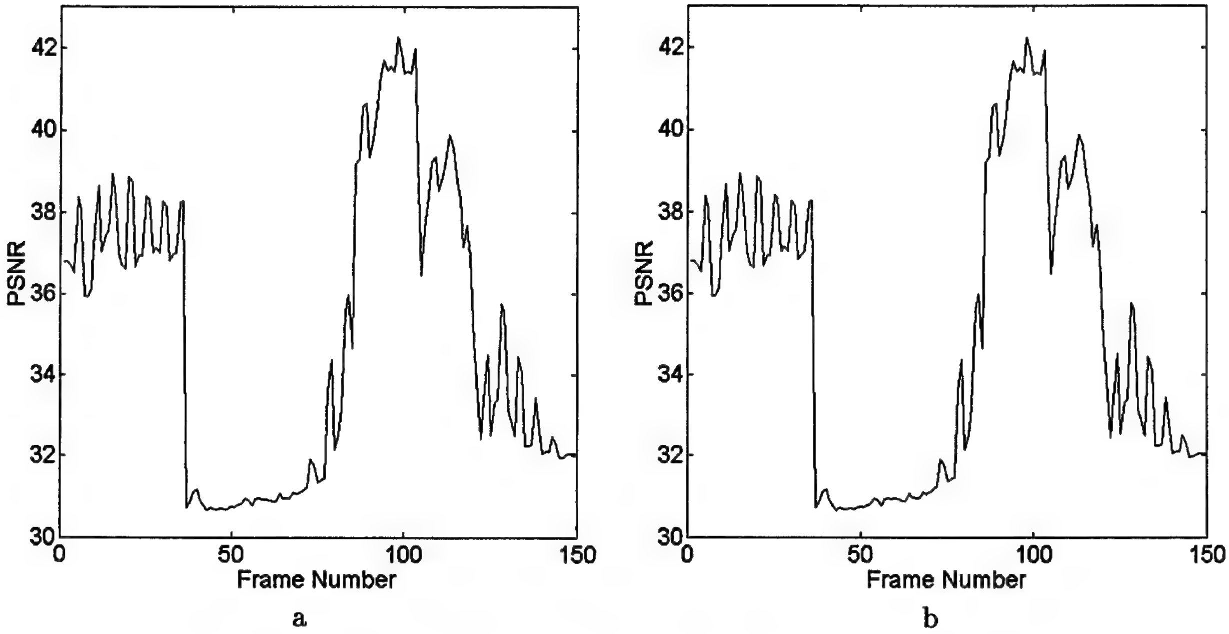


Figure 4.6. *PSNR plots with compression and audio embedding for: a) $S = 5$, $T_1 = 1.25$, $T_2 = 3.75$, and PassesOver $= 1$ and b) $S = 5$, $T_1 = 1.25$, $T_2 = 3.75$, and PassesOver $= 2$.*

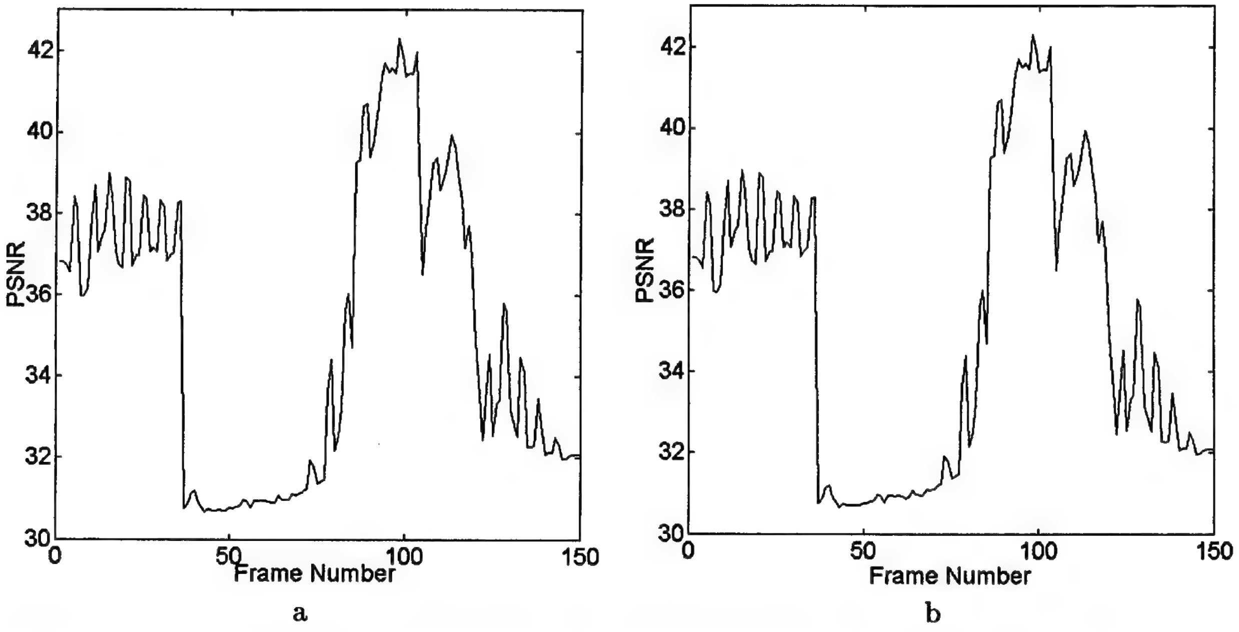


Figure 4.7. *PSNR plots with compression and audio embedding for: a) $S = 4$, $T_1 = 1$, $T_2 = 3$, and PassesOver $= 1$ and b) $S = 4$, $T_1 = 1$, $T_2 = 3$, and PassesOver $= 2$.*

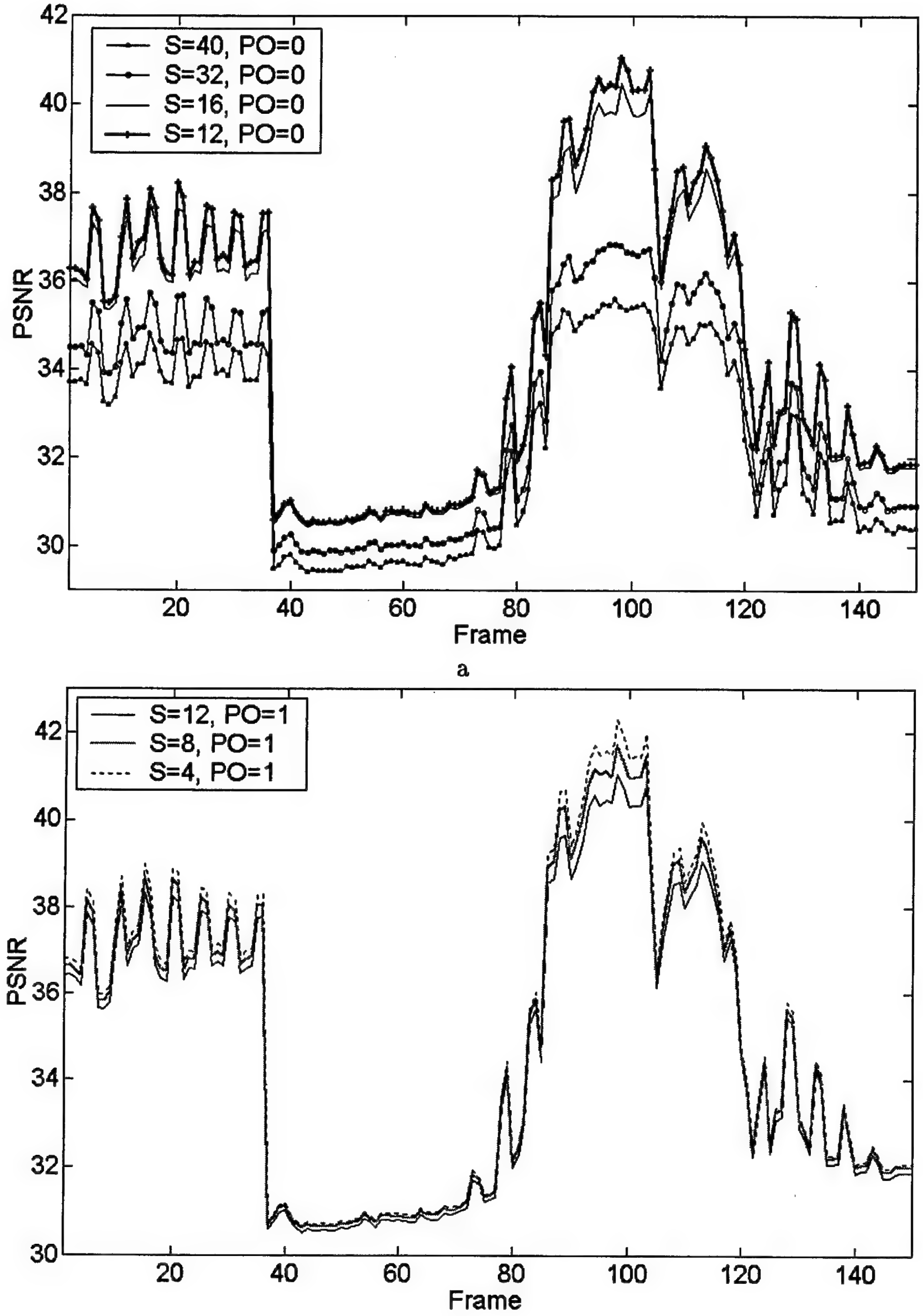
Figures 4.8a displays runs three, four, six, and eight ($S = 40$, $S = 32$, $S = 16$, and $S = 12$ respectively) in order to demonstrate that the PSNR does increase as $S$ is decreased. There is a limit to the improvement as is illustrated in 4.8b. Figure 4.8b shows runs six, eight, and nine ($S = 16$ $PO = 1$, $S = 12$ $PO = 1$, and $S = 8$ $PO = 1$.) The difference becomes less significant as $S$ is decreased and $PO$ is increased.

Table 4.3 summarizes the results of all of the data runs. As the value of $S$ is decreased, the minimum, maximum, and median PSNRs approach that of runs one and two where the wavelet coefficients were not modified with the audio information but the video data was compressed. Recall that imperceptibility and robustness conflict. As $S$ is decreased, the watermark becomes more imperceptible and the robustness is decreased. The decrease in robustness is portrayed in the value of $PO$ ($PassesOver$.) It was necessary to increase the additional number of passes in order to have enough resolution on the reconstructed coefficient to extract the audio information without errors. In other words, $PO$ is increased to compensate for the decline in robustness due to small values of $S$.

Table 4.3 also demonstrates as $S$ is decreased, compression rates generally decrease given the same value of $PO$. The highest compression of rate of 15.30:1 occurs at $S = 32$, .5270 bits per pixel, with a median PSNR of 32.83.

*4.2.2 Audio Results.* As described in Section 4.1.1, $4,270$ bits of audio data was embedded into each of the 150 video frames for a total of $640,500$ audio bits. Table 4.4 shows the number of incorrectly extracted bits. These bit errors occur when the resolution of the reconstructed value is not of high enough quality for the given value of $S$ as is the case for $S = 4$, $S = 5$, and $S = 12$. This resolution problem is solved by performing more bit-plane passes in order to refine the reconstructed values. As is shown in Table 4.4, the bit errors go away if $PO$ is incremented by one. That is, simply performing one more bit-plane coding pass alleviates the bit error problem. The downside is that compression is decreased (although overall image quality increases.)

Additionally, error control coding in the audio bit stream could be used to correct the audio bit errors. However, this would increase the size of the audio information and would require more wavelet coefficients to hold the increase in data. Increasing the number



a

b

Figure 4.8. *a) PSNR plot for* $S = 40$ $PO = 0$, $S = 32$ $PO = 0$, $S = 16$ $PO = 0$, *and* $S = 12$ $PO = 0$. *b) PSNR-vs-Frame for* $S = 16$ $PO = 1$, $S = 12$ $PO = 1$, *and* $S = 4$ $PO = 1$. *For both plots, the PSNR increases as the value of* $S$ *decreases.*

| Run | Values | Bits Per Pixel | Compression Ratio | Minimum PSNR | Maximum PSNR | Median PSNR |
|---|---|---|---|---|---|---|
| 1 | No Audio, $PO = 0$ | .5925 | 13.61:1 | 30.67dB | 42.61dB | 34.57dB |
| 2 | No Audio, $PO = 1$ | .6545 | 12.32:1 | 30.69dB | 42.63dB | 34.63dB |
| 3 | $S = 40$, $PO = 0$ | .5739 | 14.05:1 | 29.41dB | 35.62dB | 32.33dB |
| 4 | $S = 32$, $PO = 0$ | .5270 | 15.30:1 | 29.85dB | 36.87dB | 32.95dB |
| 5 | $S = 16$, $PO = 0$ | .5694 | 14.16:1 | 30.46dB | 40.46dB | 34.31dB |
| 6 | $S = 16$, $PO = 1$ | .5783 | 13.97:1 | 30.47dB | 40.46dB | 34.30dB |
| 7 | $S = 12$, $PO = 0$ | .6216 | 12.98:1 | 30.52dB | 41.04dB | 34.40dB |
| 8 | $S = 12$, $PO = 1$ | .6695 | 12.05:1 | 30.53dB | 41.02dB | 34.49dB |
| 9 | $S = 8$, $PO = 1$ | .6300 | 12.80:1 | 30.63dB | 41.69dB | 34.66dB |
| 10 | $S = 8$, $PO = 2$ | .6350 | 12.70:1 | 30.64dB | 41.68dB | 34.67dB |
| 11 | $S = 5$, $PO = 1$ | .6589 | 12.24:1 | 30.66dB | 42.25dB | 34.74dB |
| 12 | $S = 5$, $PO = 2$ | .7235 | 11.15:1 | 30.67dB | 42.20dB | 34.75dB |
| 13 | $S = 4$, $PO = 1$ | .6517 | 12.38:1 | 30.68dB | 42.28dB | 34.78dB |
| 14 | $S = 4$, $PO = 2$ | .6792 | 11.57:1 | 30.68dB | 42.28dB | 34.77dB |

Table 4.3. *Bits per pixel, compression ratio, min PSNR, max PSNR, and median PSNR for all test runs using the combined audio embedding audio/video compression technique (where $PO = PassesOver$).*

of wavelet coefficients will decrease the compression rate because the additional wavelet coefficients and their indices have to be coded.

*4.2.3 Conclusions.* Only so much gain in reconstructed video quality is reached as $S$ is decreased. In order to maintain the audio integrity (see Table 4.4) the value of $PO$ must be increased. For perfect reconstructed audio at $S = 16$, the minimum number of bit plane coding passes is acceptable. However, for perfect reconstructed audio at $S = 4$, the minimum number of bit plane coding passes must be increased by two ($PO = 2$.) There is less than .5 dB PSNR gain between these two runs (runs 5 and 14.) The cost of this PSNR gain is a 13% decrease in compression. For the input data set used, the data presented in Tables 4.3 and 4.4 show that values of $S < 16$ do not show any significant improvement in the quality of the reconstructed video sequence. Finally, the data indicates that values of $S = 32$ and $S = 16$ show very good performance.

| Run | Values | Audio Bit Errors |
|---|---|---|
| 3 | $S = 40$, $PO = 0$ | 0 |
| 4 | $S = 32$, $PO = 0$ | 0 |
| 5 | $S = 16$, $PO = 0$ | 0 |
| 6 | $S = 16$, $PO = 1$ | 0 |
| 7 | $S = 12$, $PO = 0$ | 80,121 |
| 8 | $S = 12$, $PO = 1$ | 0 |
| 9 | $S = 8$, $PO = 1$ | 0 |
| 10 | $S = 8$, $PO = 2$ | 0 |
| 11 | $S = 5$, $PO = 1$ | 160,697 |
| 12 | $S = 5$, $PO = 2$ | 0 |
| 13 | $S = 4$, $PO = 1$ | 191,845 |
| 14 | $S = 4$, $PO = 2$ | 0 |

Table 4.4. *Bit errors from extracted audio information (PO=PassesOver.)*

### *4.3 Summary*

This chapter provided information on the design and results of the experimental runs. Details on input data format was provided. Justification for parameter values used for audio embedding and audio/video compression were discussed. The compression format was solidified and details on the minimum, maximum, and median PSNR for each test run was provided. Compression rates and the average number of bits per pixel for each data run were presented. Finally, audio bit errors due to the extraction process were discussed.

# *V. Conclusions*

Through the innovative use of digital watermarking and the merging of several highly effective compression techniques, an audio embedding audio/video compression system has been developed. The resulting embedded process can be stopped at any point and the audio and video data can be reconstructed. This system can be used for the progressive transmission of the combined audio/video signal in many applications ranging from video teleconferencing to Internet multimedia.

Data presented in Tables 4.3 and 4.4 show that compression rates of 15:1 can be achieved with a median PSNR of nearly 33dB. At these rates, the audio signal can be embedded into the video signal and then extracted without audio bit errors.

Several concepts have been tied together to create this audio embedding audio/video compression system. Digital watermarking, traditionally used for copyright protection, is used in a new and exciting way to create a combined audio/video signal. This combined signal is created by modifying individual wavelet coefficients with a single bit of audio information. The watermarking technique used to embed the audio signal is efficient: it modifies the wavelet coefficients with a simple addition or subtraction of the modulus of the wavelet coefficient and a predetermined threshold parameter. Compression is achieved by exploiting the nonlinear approximation aspects of the wavelet transform. Several effective techniques are incorporated to compress the combined audio/video signal. First, bit plane coding is used to allocate bits to portions of the image where they are needed most. Second, a first-difference technique is used to code the locations of the significant wavelet coefficients. Third, Huffman coding is used to entropy code the signs of the significant coefficients and the first-difference of the coefficients indices.

## *5.1 Design and Methodology*

Four techniques for embedding an audio signal within a video signal were developed during this thesis effort. The first two methods use watermarking principles to change certain wavelet coefficients using a modular based thresholding technique developed by Tsai et al (24). These two techniques do not compress the combined audio/video signal.

The two watermarking approaches are a 2–$D$ approach and a 1–$D$ approach. The third technique encodes the audio information using a modified version of the embedded zerotree wavelet algorithm (EZW.) The fourth technique uses a modified version of the 1–$D$ audio embedding technique to embed the audio and then compresses the combined audio/video signal.

In the 2–$D$ approach, the binary audio information is treated as a binary image. That is, the audio signal is converted from a 1–$D$ signal to an $N \times N$ 2–$D$ signal. The value of $N$ is driven by the size of the subband that receives the audio data. The main advantage of this technique is that very little overhead information is needed at the decoder in order to extract the embedded audio signal. However, the main disadvantage is that it is not feasible to develop a compression technique that includes all of the wavelet coefficients in a given subband. This led to the 1–$D$ audio embedding approach.

In the 1–$D$ approach, the binary audio information is embedded in the $N$ largest wavelet coefficients (not including those coefficients that make up the coarse approximation.) The value of $N$ is driven by the amount of information that needs to be embedded in a given video frame. However, there were questions associated with this method. First, which coefficients were modified? It was the largest coefficients. However, the values of the largest coefficients where changed and may no longer ***be*** the largest coefficients. Some record of where those changed coefficients are located is required. Maintaining a record of the modified wavelet coefficients increases the overhead associated with this embedding technique. Second, what about compression? In compression techniques, two paradigms are considered: linear and nonlinear. This 1–$D$ audio embedding technique follows the nonlinear paradigm. This led to the investigation of nonlinear compression techniques.

The EZW is pursued as a compression technique due to the ability of algorithm to obtain high compression rates while maintaining high image quality. However, problems occur because modified wavelet coefficients could potentially jump threshold boundaries. It is difficult to determine how many iterations are required in order to gain enough resolution on the reconstructed wavelet coefficients to properly extract the audio signal. These problems are eliminated by encoding the audio information into one of three symbol classes: significants class, zerotree root class, and isolated zero class. This results in a larger symbol set. Although this method shows promise, the EZW algorithm is very inefficient, and does not satisfy the *real-time processing* requirements of this thesis effort.

The final technique explored in this thesis effort achieves audio embedding and audio/video compression through the use of several key concepts. First, digital watermarking is used to embed the audio data into the $N$ largest magnitude wavelet coefficients. Unlike the 1–$D$ audio embedding technique, the coarse approximation is modified with audio data. Second, several effective compression techniques are combined to compress the combined audio/video signal. These methods include: bit plane coding to allocate bits where they are needed most, index coding to code the signs and indices of the modified wavelet coefficients, and Huffman coding to code the symbols resulting from index coding.

This audio embedding audio/video compression system is not "real-time" as implemented in Matlab. The complexity of the algorithm revolves around the implementation of the discrete wavelet transform (DWT) and obtaining the $N$ largest magnitude wavelet coefficients. However, by implementing the DWT and the extraction portion of the algorithm with an applications specific integrated circuit (ASIC) or a floating point gate array (FPGA), this audio embedding audio/video compression system could be implemented in real-time.

### *5.2 Results of Experiments*

Experiments show that the audio embedding audio/video compression system developed during this research has excellent results. The audio information can be embedded and then extracted without audio bit errors providing the resolution of the reconstructed wavelet coefficients is adequate. Overall compression rates as high as 15:1 are achieved. The reconstructed video signal has a median PSNR of nearly 33 dB. Results show that the reconstructed video quality improves as the value of $S$ is decreased. However, there is a limit to this improvement. Experimental results show that values of $S < 16$ show little improvement in video quality, reduce the compression ratio, and increase the chance of audio bit errors. These audio bit errors are eliminated by performing additional bit plane passes in order to increase the resolution of the reconstructed wavelet coefficients.

### *5.3 Future Work*

This thesis effort focuses on the embedding of the audio data and the compression of the combined audio/video signal. The next step is to pre-process the audio signal before the embedding process. There are two areas of investigation: progressive audio coding (as with the bit plane coding of significants of each video frame) and high-quality audio compression.

Progressive transmission of audio data should be accomplish on a frame-by-frame basis. This allows for a progressive audio/video sequence. There are several issues to consider: the use of wavelet packets (best basis selection) versus the "standard" wavelet transform, how to include the best basis selection, and how to progressively code non-monaural sound (i.e., stereo, Dolby ProLogic, Dolby Digital, etc.)

High-quality audio compression is desirable for one of two reasons. First, it might be desirable to embed high-quality audio signals, such as a 16-bit stereo signal, into a video signal. Second, it might be desirable to incorporate some sort of error control coding using redundant audio information. In order to incorporate error control coding, it would be necessary to compress the audio signal in order to limit the amount of information that needs to be coded. Two excellent resources on wavelet-based audio compression are: Srinivasan et al (21) and Sinha et al (19). These audio compression methods are described in Appendix E.2.1.

### *5.4 Contributions*

The audio embedding audio/video compression algorithm developed in this research is generic. It can be used to embed any binary representation of a $1\text{–}D$ signal into a 1 or $2\text{–}D$ signal in the wavelet domain. Additionally, the algorithm is efficient (in terms of big oh notation):

1. Discrete Wavelet Transform – This is an $O(n)$ operation where $n$ is the number of pixels in each video frame.
2. Extracting the $N$ largest wavelet coefficients – This can be accomplished by first sorting the data in descending order and extracting elements 1-$N$. This sorting can

be accomplished in $O(n \log n)$ where $n$ is the number of wavelet coefficients (which is equal to the number of pixels in the video frame.)

3. Embedding/Compression – This can be accomplished in $O(N)$ where $N$ is the number of extracted wavelet coefficients.

This algorithm has a wide range of uses. It can be used in an image, audio, or video database retrieval system. In this particular application, the image, audio, or video would be stored in its compressed representation. The watermark would be used to describe the contents of the image, audio, or video information. Another application is to embed text in images, text in audio, or text in video. The textual information can be used to provide detailed information regarding the compressed media.

Finally, the algorithm developed in this research shows great promise. It is able to compress the video data by a factor of 15:1, a file size reduction of 93%. The reconstructed video quality has a median PSNR of nearly 33dB. Additionally, the embedded audio information can be extracted without error. The applications of this embedding/compression system are limited only by the imagination.

# *Appendix A. Biorthogonal Wavelet Transform*

## *A.1 Generalizing to Biorthogonal Wavelets*

In *Chapter 2*, wavelets were developed based on an orthogonal decomposition of some function $f$ into the subspaces $V_m$ and $W_m$. As such, the lowpass and highpass filters ($h$ and $g$, respectively) for the inverse DWT are the same as the lowpass and highpass filters of the DWT. By loosening the the orthogonality requirement, a biorthogonal wavelet transform can be created (4, 25). This is accomplished by using *dual spaces.* The dual spaces are $\widetilde{V}_m$ and $\widetilde{W}_m$. In the biorthogonal spaces, $V_m = \textbf{Span}\{\phi_{m,k}\}$, $\widetilde{V}_m = \textbf{Span}\{\widetilde{\phi}_{m,k}\}$, $W_m = \textbf{Span}\{\psi_{m,k}\}$, and $\widetilde{W}_m = \textbf{Span}\{\widetilde{\psi}_{m,k}\}$. In order for the dual space to be biorthogonal, $V_m \cap \widetilde{W}_m = \emptyset$, $\widetilde{V}_m \cap W_m = \emptyset$, $V_m \perp \widetilde{W}_m$ and $\widetilde{V}_m \perp W_m$, and the basis functions must satisfy the *biorthogonality condition.* The biorthogonality condition is:

$$\left\langle \phi_{m,k}, \widetilde{\phi}_{m,l} \right\rangle = \delta(k-1) \tag{A.1}$$

$$\left\langle \psi_{m,k}, \widetilde{\psi}_{m,l} \right\rangle = \delta(k-1). \tag{A.2}$$

In order to illustrate the idea of biorthogonality, Figure A.1 shows a decomposition of space $V_0 = \Re^3$. The space $V_1$ is spanned by $\phi_0$ and $\phi_1$. The space $\widetilde{V}_1$ is spanned by $\widetilde{\psi}_0$ and $\widetilde{\psi}_1$. As such, $V_1$ and $W_1$ are orthogonal to their dual spaces, $\widetilde{W}_1$ and $\widetilde{V}_1$ respectively. Note that these basis functions satisfy Equations A.1 and A.2, the biorthogonality conditions.

The recursion relations are the same as before. That is,

$$\begin{aligned} \phi_{m,l}(x) &= \sum_k h_{l,k}\, \phi_{m-1,k}(x), \\ \psi_{m,l}(x) &= \sum_k g_{l,k}\, \phi_{m-1,k}(x). \end{aligned}$$

In order to find the lowpass filter $h$, the inner product of $\phi_{m,l}$ is taken with $\widetilde{\phi}_{m,l}$ (its dual function.) This yields:

$$\begin{aligned} \left\langle \phi_{m,l}, \widetilde{\phi}_{m-1,l} \right\rangle &= \sum_n h_{l,n} \left\langle \phi_{m-1,n}, \widetilde{\phi}_{m-1,k} \right\rangle \\ &= \sum_n h_{l,n}\, \delta(k-n) \end{aligned}$$

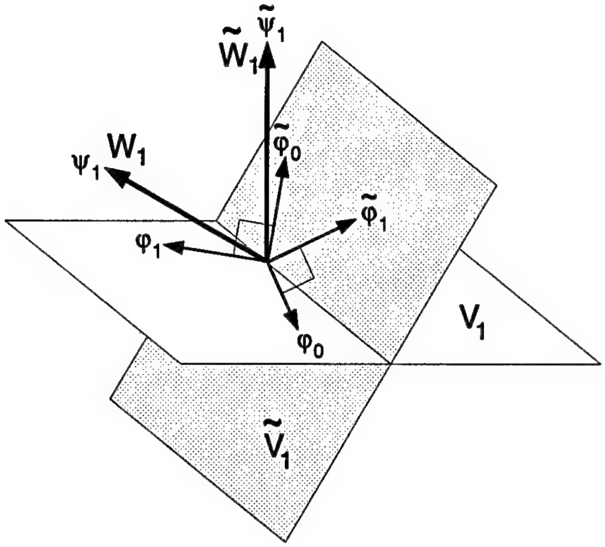


Figure A.1. Decomposition of $V_0 = \Re^3$.

$$= \quad h_{l,k}. \tag{A.3}$$

Similarly, to find the highpass filter $g$, the inner product of $\psi_{m,l}$ is taken with $\widetilde{\phi}_{m,l}$ (its dual function.) This yields:

$$\begin{aligned}
\left\langle \psi_{m,l}, \widetilde{\phi}_{m-1,l} \right\rangle &= \sum_n g_{l,n} \left\langle \psi_{m-1,n}, \widetilde{\phi}_{m-1,k} \right\rangle \\
&= \sum_n g_{l,n} \, \delta(k-n) \\
&= g_{l,k}.
\end{aligned} \tag{A.4}$$

Similarly, the equivalent recursion relations for $\widetilde{h}$ and $\widetilde{g}$ are determined.

### *A.2 Constructing the Biorthogonal Discrete Wavelet Transform*

With these dual spaces, the function $f$ (which exists in the space $V_{m-1}$) can be decomposed. The decomposition of $f$ into $V_{m-1}$ is:

$$f(x) = \sum_k c_{m-1,k} \, \widetilde{\phi}_{m-1,k}(x)$$

However, the following decomposition of $f(x)$ is desired:

$$f(x) = \sum_k c_{m,k}\, \tilde{\phi}_{m,k}(x) + \sum_k d_{m,k}\, \tilde{\psi}_{m,k}(x).$$

Recall from Section 2.3.3 that the scaling coefficients $c_{m,k}$ are found by taking the inner product of the function $f$ with $\phi_{m,k}$. That is,

$$\begin{aligned} c_{m,k} &= \langle f, \phi_{m,k} \rangle \\ &= \sum_k c_{m-1,l} \left\langle \phi_{m,k}, \tilde{\phi}_{m-1,l} \right\rangle \\ &= \sum_k c_{m-1,l}\, h_{k,l}. \end{aligned} \tag{A.5}$$

Similarly, the wavelet coefficients $d_{m,k}$ are found by taking the inner product of $f$ with $\psi_{m,k}$

$$\begin{aligned} d_{m,k} &= \langle f, \psi_{m,k} \rangle \\ &= \sum_k d_{m-1,l} \left\langle \phi_{m,k}, \tilde{\psi}_{m-1,l} \right\rangle \\ &= \sum_k d_{m-1,l}\, g_{k,l}. \end{aligned} \tag{A.6}$$

In the above equations, the lowpass filter $h$ and the highpass filter $g$ are independent of scale. Equations A.5 A.6 determine the forward biorthogonal wavelet transform. Note that the reconstruction equations are the same for the orthogonal and biorthogonal cases.

### *A.3 Reconstruction (Biorthogonal Discrete Wavelet Transform)*

This section provides the details to reconstruct the signal $f$ using the biorthogonal wavelet transform. The function $f$ can be described by the biorthogonal wavelet coefficients at scale $m$ as:

$$f(x) = \sum_k c_{m,k}\, \tilde{\phi}_{m,k}(x) + \sum_k d_{m,k}\, \tilde{\psi}_{m,k}(x). \tag{A.7}$$

The function $f$ exists in $V_{m-1}$ and is represented as:

$$f(x) \;=\; \sum_k c_{m-1,k}\, \tilde{\phi}_{m1,k}(x). \tag{A.8}$$

If the inner product of Equations A.7 and A.8 are taken with $\phi_{m-1,l}$, this results in the following:

$$\begin{aligned}\langle f, \phi_{m-1,l}\rangle &= \sum_k c_{m-1,k} \left\langle \tilde{\phi}_{m-1,k}, \phi_{m-1,l} \right\rangle \\ &= c_{m-1,l} \\ &= \sum_k c_{m,k} \left\langle \tilde{\phi}_{m,k}, \phi_{m-1,l} \right\rangle + \sum_k d_{m,k} \left\langle \tilde{\psi}_{m,k}, \phi_{m-1,l} \right\rangle \\ &= \sum_k c_{m,k}\, \tilde{h}_{k,l} + \sum_k d_{m,k}\, \tilde{g}_{k,l}. \end{aligned} \tag{A.9}$$

Notice that the reconstruction for the orthogonal (Equation 2.14) and biorthogonal (Equation A.9) cases are the same with the exception of the filters. The orthogonal case uses $h$ and $g$ where as the biorthogonal case uses the dual filters $\tilde{h}$ and $\tilde{g}$. Furthermore, Equations A.5, A.6, and A.9 lead to a filter bank implementation of the biorthogonal discrete wavelet transform.

# *Appendix B. Importance of Linear Phase*

Recall that linear phase filters are desirable because of the importance of phase in reconstructing an image. As an example, consider the 256 × 256 pixel "Lenna" image in Figure B.1. Figure B.2 shows the same "Lenna" image with magnitude information only. The magnitude information is obtained from the Fourier transform of the origninal "Lenna" image. The inverse Fourier transform is performed on the magnitude of the Fourier coefficients only. That is, phase information is not included. Figures B.3 and B.4 show the "Lenna" image with phase information only. The phase information is obtained from the Fourier transform of the orginal "Lenna" image. Figure B.3 displays the reconstructed image where the magnitudes of the original Fourier coefficients are replaced with a constant magnitude of one. Figure B.4 displays the reconstructed image where the magnitudes of the Fourier coefficients are replaced with a normally distributed random magnitude ($N(0, 1)$.) As can be seen from this example, maintaining the integrity of the phase information within an image is very important. Therefore, when processing images, using filters that have linear phase is extremely important. This why biorthogonal filters (and not orthogonal filters) are used in the discrete wavelet transform for this thesis.

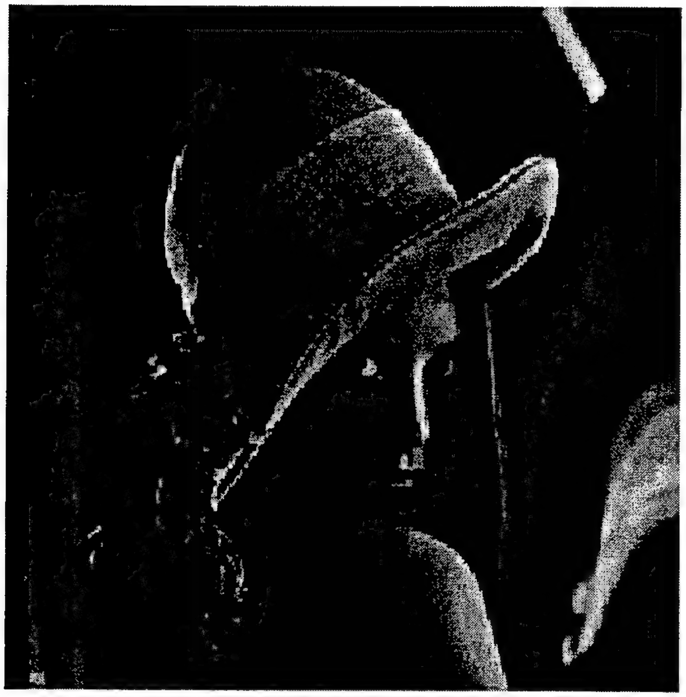


Figure B.1. *Original* 256 × 256 *pixel* 8-*bit grayscale "Lenna" image.*

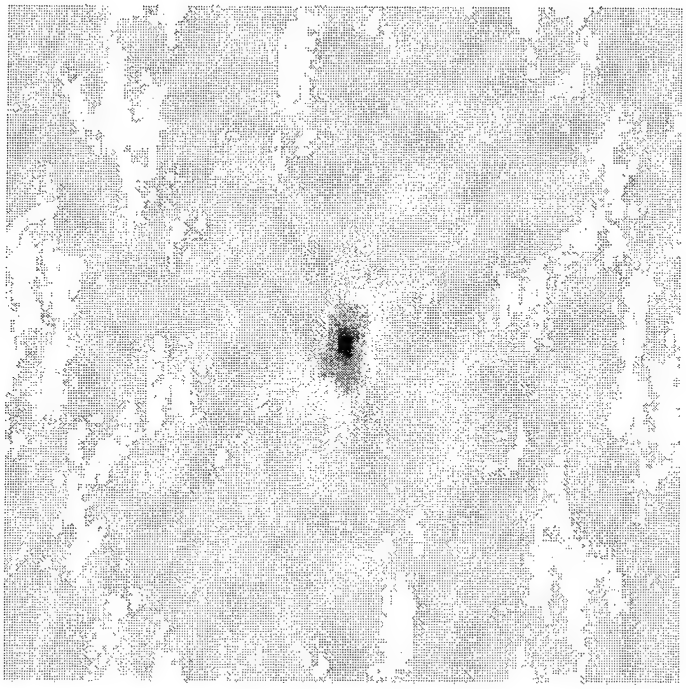


Figure B.2. *Original* 256 × 256 *pixel* 8-*bit grayscale "Lenna" image reconstructed using magnitude information only. This image is the reverse image.*

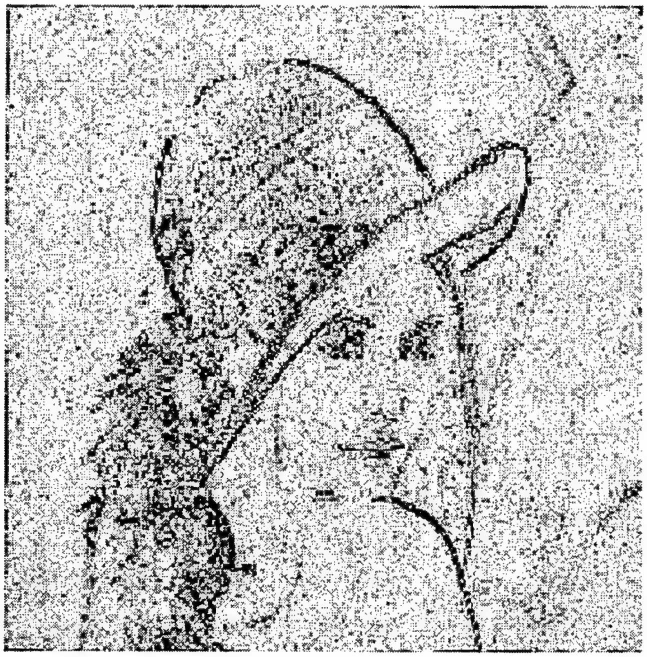


Figure B.3. *Original* 256 × 256 *pixel* 8-*bit grayscale "Lenna" image reconstructed using phase information and constant magnitude of one. This image is the reverse image.*

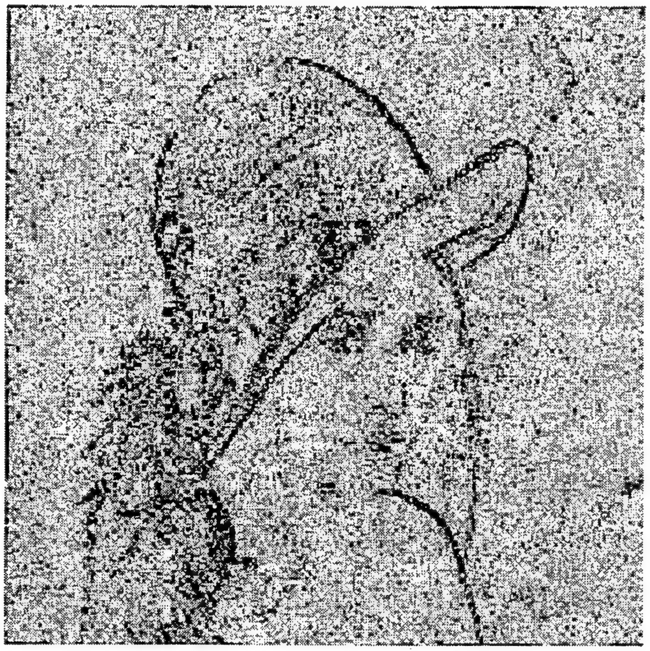


Figure B.4. *Original* 256 × 256 *pixel* 8-*bit grayscale "Lenna" image reconstructed using phase information and a normally distributed* ($N(0,1)$) *random magnitude. This image is the reverse image.*

# *Appendix C. Embedded Zerotree Wavelet Compression*

## *C.1 EZW Compression Example*

1. $S = \{\emptyset\}$, $D = \{\emptyset\}$
2. Pass 1

   (a) Search to find the maximum magnitude and set the initial threshold $T_{init}$: max value $= -116$, therefore $T_{init} = 2^{\lfloor \log_2(|-116|) \rfloor} = 2^6 = 64$

   (b) Perform a dominant pass to determine the significant values vector and the symbol stream vector:

   $$S_1 = \{70 \;\; -116 \; 96 \; 81\}, \text{ set}$$
   $$S = S \cup S_1 = \{\emptyset\} \cup \{70 \;\; -116 \; 96 \; 81\} = \{70 \;\; -116 \; 96 \; 81\}$$

   $D_1 = \{\text{PS NS PS PS}\}$, set $D = D \cup D_1 = \{\emptyset\} \cup \{\text{PS NS PS PS}\} = \{\text{PS NS PS PS }\}$

   (c) The values in the significants vector $S$ are compared with the odd multiples of $\frac{T_{crnt}}{2}$ ($\frac{k_{odd} * T_{crnt}}{2}$ where $k \in \mathcal{Z}$) that lie between $T_{crnt}$ and $2 * T_{init}$. $T_{crnt}$ is the current threshold. Because these are new values (there are no significants to refine), we compare the current set of new significant with $\frac{3 * T_{crnt}}{2} = \frac{3 * 64}{2} = 96$. If the significant is greater than or equal to $\frac{3 * T_{crnt}}{2}$, then it is coded as a 1, otherwise it is coded as a 0 (see Table C.1. $C$ denotes the comparison value(s) and *Bit* denotes the bit value to encode the significant.)

| $T_{crnt}$ | Bit< | C | ≤Bit | $2 * T_{init}$ |
|---|---|---|---|---|
| 64 | 0 | 96 | 1 | 128 |

Table C.1. *EZW Compression: Pass 1 bit table for new values with* $T = 64$.

   (d) The resulting bit stream is:

   $$B = B \cup B_1 = \{\emptyset\} \cup \{0110\} = \{0110\}$$

   (e) Because this is the first Dominant/Subordinate Pass, there are no values to refine.

3. Pass 2

   (a) Determine the new threshold value which is equal to half the previous threshold value:

$$T_{crnt} = \frac{T_{init}}{2} = \frac{64}{2} = 32$$

   (b) Perform the dominant pass to get the significant values and the symbols:

$$S_2 = \{32\ 55\ 48\},$$

$$S = S \cup S_2 = \{70\ 116\ 96\ 81\} \cup \{32\ 55\ 48\} = \{70\ 116\ 96\ 81\ 32\ 55\ 48\},$$

$$D_2 = \{\text{PS PS PS }\},\ D = D \cup D_2 = \{\text{PS NS PS PS}\} \cup \{\text{PS PS PS}\} = \{\text{PS NS PS PS PS PS PS}\}.$$

   (c) The values in the significants vector found during the current iteration (i.e. $S_2$) are compared with the odd multiples of the current threshold divided by two $\frac{k_{odd}*T_{crnt}}{2}$.

$$\frac{3*T}{2} = \frac{3*32}{2} = 48$$

   If the significant is greater than or equal to 48, then it is coded as a 1, otherwise it is coded as a 0 (see Table C.2.)

   (d) The old significant values must now be refined (these are those values in $S$ that are not in $S_1$.) This is accomplished by using the previous bit value and comparing the significant with odd multiples of $\frac{T_{crnt}}{2}$ (the set of integer values used to compare with the significants are determined by the number of iterations performed.) If the significant is greater then or equal to the comparison value, then it is refined with a 1, otherwise it is refined with a 0 (see Table C.3. $C$ denotes the comparison value(s) and *Bit* denotes the bit value to encode the significant.)

| Used to Code New Values | | | | |
|---|---|---|---|---|
| $T_{crnt}$ | Bit< | C= $\frac{3*T_{crnt}}{2}$ | ≤ Bit | $2 * T_{crnt}$ |
| 32 | 0 | 48 | 1 | 64 |

Table C.2. *EZW Compression: Pass 2 bit table for new values with* $T = 32$.

| Used to Refine Old Values | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Previous Bit = 0 | | | | | Previous Bit = 1 | | | |
| $2 * T_{crnt}$ | Bit< | C= $\frac{5*T_{crnt}}{2}$ | ≤Bit | $\frac{6*T_{crnt}}{2}$ | Bit< | C= $\frac{7*T_{crnt}}{2}$ | ≤Bit | $2 * T_{init}$ |
| 64 | 0 | 80 | 1 | 96 | 0 | 112 | 1 | 128 |

Table C.3. *EZW compression: Pass 2 bit table for refining old values with $T = 32$.*

| $B_2$ | $B_1$ | Resulting $B$ |
|---|---|---|
| (New)∪(Refined) | (Original) | (Original)∪(New)∪(Refined) |
| {(011)} ∪ {(0101)} | {(0110)} | {(0110)(011)(0101)} |

Table C.4. *EZW Compression: Final bit stream.*

(e) The resulting bit stream is: {01100110101} (see Table C.4).

## *C.2 EZW Reconstruction Example*

1. Pass 1

(a) The values in the bit stream vector $B$ created during the subordinate passes are used in conjunction with the current threshold $T_{crnt} = T_{init}$ to determine the reconstruction values for the positive and negative significants. Recall from the compression example that the initial threshold $T_{init} = 64$. Also recall from the compression example above that the bits of $B$ that pertain to the current pass are {0110}. Table C.5 is then used to determine the reconstruction values. $C$ denotes the comparison value(s), $R$ denotes the reconstruction values, and *Bit* denotes the bit value to encode the significant

| | Bit=0 | | Bit = 1 | |
|---|---|---|---|---|
| $T_{crnt}$ | R= $\frac{5*T_{crnt}}{4}$ | C | R= $\frac{7*T_{crnt}}{4}$ | $2 * T_{init}$ |
| 64 | 80 | 96 | 112 | 128 |

Table C.5. *EZW Reconstruction: Pass 1 reconstruction levels for new bits with $T = 32$.*

(b) The reconstructed significant values are then:

{80 112 112 80}

(c) After the significants have been reconstructed, their sign is determined by the symbol stream vector. If the symbol is a PS, then the value remains positive. If the symbol is a NS, then the reconstruction value for that significant is negated. Recall the symbols that pertain to the current pass are:

{PS NS PS PS }

The resulting reconstructed values for this first pass are:

$$\{80\ -112\ 112\ 80\}$$

2. Pass 2

(a) During this pass of the reconstruction phase, it is necessary to do two things. First, determine the reconstruction values of the new set of bits. Second, refine the reconstruction values of previously reconstructed values using the refinement bits.

(b) The values in the bit stream vector $B$ created during the subordinate passes are used in conjunction with the current threshold $T_{crnt} = \frac{T_{crnt}}{2} = \frac{64}{2} = 32$ to determine the reconstruction values for the positive and negative significants. Recall from the compression example above that the bits of $B$ that pertain to the current pass are $\{(011)(0101)\}$ .

(c) The first set of bits ($\{011\}$) are used to determine the reconstruction values of the set of new coefficients. Table C.6 is then used to determine the reconstruction values for the new set of bits. $C$ denotes the comparison value(s), $R$ denotes the reconstruction values, and *Bit* denotes the bit value to encode the significant.

|  | Bit=0 |  | Bit = 1 |  |
|---|---|---|---|---|
| $T_{crnt}$ | R= $\frac{5*T_{crnt}}{4}$ | C | R= $\frac{7*T_{crnt}}{4}$ | $2 * T_{crnt}$ |
| 32 | 40 | 48 | 56 | 64 |

Table C.6. *EZW Reconstruction: Pass 2 reconstruction levels for new bits with* $T = 32$.

(d) The new reconstructed values are: {40 56 56}

(e) The second set of bits ({0101}) are used to refine the existing coefficients from the previous pass. Table C.7 is used to determine the reconstruction values for the new set of bits.

| $2 * T_{crnt}$ | PBit=0 | | | C | PBit=1 | | | $2 * T_{init}$ |
|---|---|---|---|---|---|---|---|---|
| | CBit=0<br>R= $\frac{9 * T_{crnt}}{4}$ | C | CBit = 1<br>R= $\frac{11 * T_{crnt}}{4}$ | | CBit=0<br>R= $\frac{13 * T_{crnt}}{4}$ | C | CBit=1<br>R= $\frac{15 * T_{crnt}}{4}$ | |
| 64 | 72 | 80 | 88 | 96 | 104 | 112 | 120 | 128 |

Table C.7. *EZW Reconstruction: Pass 2 reconstruction value for refinement bits with* $T = 32$.

i. C denotes the comparison value(s).

ii. R denotes the reconstruction values.

iii. CBit denotes the current bit value to encode the significant.

iv. PBit denotes the previous bit value to encode the significant.

(f) The resulting refined significants are:

$$\{72 \ -120 \ 112 \ 80\}$$

(g) The full list of reconstructed values are:

$$\{72 \ -120 \ 112 \ 80 \ 40 \ 56 \ 56\}$$

# *Appendix D. Entropy Coding*

## *D.1 Huffman Coding Example*

This section provides a simple example of how to Huffman code a given set of symbols. Consider the following symbols and probabilities:

| | | |
|---|---|---|
| ’*A*’ | - | .3 |
| ’1’ | - | .05 |
| ’*B*’ | - | .2 |
| ’2’ | - | .2 |
| ’?’ | - | .25 |

The Huffman coding process starts by ordering the symbols in decreasing order of probability. That is:

| | | |
|---|---|---|
| ’*A*’ | - | .3 |
| ’?’ | - | .25 |
| ’*B*’ | - | .2 |
| ’2’ | - | .2 |
| ’1’ | - | .05 |

The next step is to start from the bottom and add the two lowest probabilities together as in Figure D.1. These two symbols are combined onto a single branch, which is assigned the new probability. A 1 is assigned to the top branch and a 0 to the bottom branch.

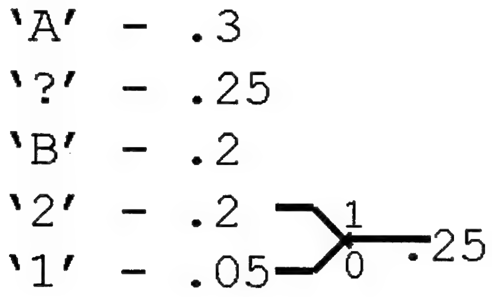


Figure D.1. *Huffman coding: adding the two smallest probabilities together and combinning them into a single branch is the start of generating the Huffman code.*

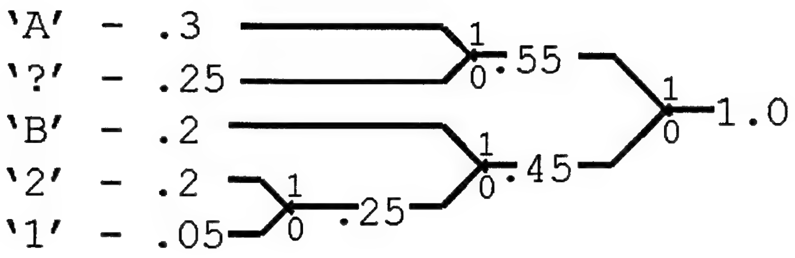


Figure D.2. *Huffman coding: all probabilities added together to generate the Huffman code.*

This process is continued until all probabilities are accounted for as in Figure D.2. In order to determine the bit representation of each symbol, the 1's and 0's are traced from the far right to the symbol on the left while writing them down from left to right. Doing so yields the following Huffman code words:

$$\begin{aligned} '\mathit{A}' &= 11 \\ '?' &= 10 \\ '\mathit{B}' &= 01 \\ '2' &= 001 \\ '1' &= 000 \end{aligned}$$

To demonstrate the effectiveness of Huffman coding, it is necessary to compare the average number of bits per symbol (bps) between the Huffman solution, the entropy $H$ (which is the theoretical minimum), and a uniform length code word:

Huffman: $(.3 * 2 + .25 * 2 + .2 * 2 + .2 * 3 + .05 * 3) = 2.25$ bps

Entropy: $(.3 * \log_2 .3 + .25 * \log_2 .25 + .2 * \log_2 .2 + .2 * \log_2 .2 + .05 * \log_2 .05) = 2.0235$ bps

Uniform Length: 3 bps

By using using Huffman coding in the above example, it is evident that the average number of bits required to code each symbol is close to that of the entropy $H$ and much less than using a single length code word.

# *Appendix E. Audio Compression*

## *E.1 Acoustic Models*

Acoustic models describe the characteristics of the human auditory system (HAS.) Sound processing systems are designed using aspects of acoustic models in order to take advantage of known properties exhibited by HAS. Auditory masking phenomenon are summarized below:

1. Frequency masking – Occurs when a weak auditory signal is masked (overrun or overpowered) by a strong auditory signal.
2. Temporal masking – Temporal masking encompasses two different constructs: post-masking and pre-masking.
   - (a) Post-masking (backward masking) – Occur when a strong auditory signal masks a weak signal where the weak signal occurs after the strong signal (9, 22).
   - (b) Pre-masking (forward masking) – Occur when a strong auditory signal masks a weak signal where the weak signal occurs before the strong signal (9, 22).
3. Perceptual Entropy – The limit for which an audio signal can be compressed such that the perceived distortion is zero.

## *E.2 Audio Compression*

The concept of audio compression is straightforward: reduce the amount of audio information without significantly degrading the quality of the audio signal. There are many reasons to compress an audio signal. One reason is to reduce the amount of storage space required for the audio signal. A more common but related reason is the increasing desire for high-quality audio signals over the Internet.

*E.2.1 Multiresolution Techniques.* A low bit rate audio compression technique based on two key concepts is presented in (19). First, this technique takes advantage of the masking characteristics of the human hearing process by using an optimal wavelet decomposition technique called adapted wavelets (or wavelet packets). Adapted wavelets optimally select the discrete wavelet transform (DWT) with potential decompositions of both the lowpass and highpass subbands, of each audio frame. Second, this technique leverages statistical redundancies within the audio signal via dynamic dictionary based encoding. Combining the wavelet packet and dynamic dictionary approaches, this technique was able to reduce significantly the amount of information contained in the compressed audio signal. The sample high-fidelity audio signal required 705 Kbits of data for one second of audio information. This compression technique allowed the authors to reduce the compressed signal to approximately $48 - 66$ Kbits for one second of audio information. Because of a "long coding delay" (19), this technique is not suitable for real-time audio compression. The audio information can be decoded in real-time making this technique suitable for Internet-based multimedia applications.

In (21), an adaptive wavelet technique for compressing high-quality audio information is presented. The adaptive wavelet filter bank changes to match the audio signal according to the psychoacoustic model. The filter adapts such that the compressed audio signal approaches the perceptual entropy (PE) and is defined as the limit the audio signal can be compressed to achieve a perceived distortion of zero. In order to compress high-quality audio signals, this technique takes advantage of two specific aspects of the masking characteristic of the human hearing system: efficient bit allocation and progressive transmission. The zero tree wavelet algorithm developed in (18) is used for progressive transmission. This technique lends itself to progressive real time encoding and decoding.

# *Appendix F. Algorithm Functional Descriptions*

## *F.1 2–D Extraction*

1. Description: The algorithm below defines a method to embed the audio information into various wavelet coefficients. This algorithm is iterative.

2. Data Dictionary:

   (a) $I^{mod}$ is the input image modified with the watermark.

   (b) $I^{mean}$ is the mean of the modified image.

   (c) $I^{temp}$ is a temporary variable used to store a modified version of $I^{mod}$.

   (d) $W^{mod}$ is the set of wavelet coefficients modified with the data in $A$.

   (e) $E^{mod}$: $E^{mod} \subset W^{mod}$. That is, $E^{mod}$ is the set of wavelet coefficients that contain the watermark information.

   (f) $A^{recon}$ is the binary audio data that is extracted from $I^{mod}$.

   (g) $S$ is an integer value used for a modulus operation. As $S$ increases, the imperceptibility of the watermark decreases, but the robustness of the watermark increases. As $S$ decreases, the imperceptibility of the watermark increases, but the robustness of the watermark decreases.

   (h) $(p_1, p_2)$ is the insertion point of the $N \times N$ audio "image."

   (i) $N$ is the size of the audio image ($N \times N$.)

   (j) $T_1$ is a threshold value used to embed the watermark into $I$.

   (k) $T_2$ is a threshold value used to embed the watermark into $I$.

   (l) $a$ is an index counter for the outer for loop.

   (m) $b$ is an index counter for the inner for loop.

   (n) $index_{audio}$ is an index to the current location within the array of audio coefficients.

   (o) $index_{video}$ is an index to the current location within the array of video frames.

3. Functional Description:

(a) $index_{audio} = 1$.

(b) $index_{video} = 1$.

(c) $S = 32$: Work shown by Tsai et al indicates that this is a good value for $S$ (24).

(d) $T_1 = \frac{3*S}{4} = 24$: Work shown by Tsai et al indicates that this is a good value for $T_1$ (24).

(e) $T_2 = \frac{S}{4} = 8$: Work shown by Tsai et al indicates that this is a good value for $T_2$ (24).

(f) *Init*:

- i. if $index_{video} > length(I)$,
  then return,
  else continue.
- ii. $I^{mean} = mean(I^{mod}(index_{video}))$: Get the mean of the current frame.
- iii. $I^{temp} = I^{mod}(index_{video}) - I^{mean}$: Zero mean the current frame.
- iv. $W = \mathcal{DWT}(I^{temp}, L)$: Choose $L = 6$ to perform 6 iterations of the $\mathcal{DWT}$.
- v. $E$ equals an $N \times N$ sub-matrix of $W$ starting at some point $(p_1, p_2)$ i.e. $E = W(p_1 : p_1 + N, p_2 : p_2 + N)$. $E$ is typically a given subband of at a given scale within $W$. For example, the LH subband of scale 5 of the $\mathcal{DWT}$ performed 6 times on a $256 \times 256$ pixel image would be a $128 \times 128$ pixel area consisting of $128^2 = 16,384$ elements.

(g) *Extract*:

- i. for $a = 1$ to $N$.
- ii. for $b = b$ to $N$.
- iii. if $index_{audio} > length(A)$,
  then GoTo *Post Processing*,
  else continue.
- iv. if $(|W^{mod}| \bmod S) \geq \frac{T_1+T_2}{2}$,
  then $A^{recon}(index_{audio}) = 0$.

        v. else $(|W^{mod}| \bmod S) < \frac{T_1+T_2}{2}$,
           then $A^{recon}(index_{audio}) = 1$.
        vi. $index_{audio} = index_{audio} + 1$.
        vii. $index_{video} = index_{video} + 1$.

    (h) *Post Processing*:

        i. $I^{mod}(index_{video}) = \mathcal{DWT}^{-1}(W^{mod}, L)$: Perform the inverse discrete wavelet transform on $W^{mod}$.
        ii. $I^{mod}(index_{video}) = I^{mod}(index_{video}) + I^{mean}$: Add the mean back into the image.
        iii. $I^{mod}(index_{video}) = quantize(I^{mod}(index_{video}))$: Quantize the pixels of the modified frame.
        iv. $index_{video} = index_{video} + 1$: Increment the video frame array index.
        v. GoTo *Init*

### *F.2 1-D Embedding*

1. Description: The algorithm below defines a method to embed the audio information into various wavelet coefficients. This algorithm is iterative.

2. Data Dictionary:

    (a) $I$ is the set of input frames.

    (b) $I^{mean}$ is the mean of the current frame.

    (c) $I^{temp}$ is a temporary variable used to store a modified version of the current frame.

    (d) $I^{mod}$ is the set of video frames embedded with the watermark information.

    (e) $W$: $W = \mathcal{DWT}(I^{temp})$. Note: $\mathcal{DWT}(I^{temp}, L)$ indicates that $L$ iterations of the $\mathcal{DWT}$ should be performed.

    (f) $A$ is the binary audio data that is embedded as a watermark within $I$.

    (g) $W^{mod}$ is the set of wavelet coefficients modified with the data in $A$.

(h) $E$: $E \subset W$. $E$ is the set of wavelet coefficients that receive the watermark information in $A$.

(i) $E^{mod}$: $E^{mod} = \mathcal{F}(E, A)$. Here, $\mathcal{F}(E, A)$ is the embedding process defined Section 3.2.

(j) $S$ is an integer value used for a modulus operation. As $S$ increases, the imperceptibility of the watermark decreases, but the robustness of the watermark increases. As $S$ decreases, the imperceptibility of the watermark increases, but the robustness of the watermark decreases.

(k) $P$ is an array that maintains the indices to the locations of the largest wavelet coefficients

(l) $T_1$ is a threshold value used to embed the watermark into $I$.

(m) $T_2$ is a threshold value used to embed the watermark into $I$.

(n) $a$ is an index counter for the main *for loop.*

(o) $index_{audio}$ is an index to the current location within the array of audio coefficients.

(p) $index_{video}$ is an index to the current location within the array of video frames.

3. Functional Description: Describes the iterative algorithm used to embed the watermark.

(a) $index_{audio} = 1$

(b) $index_{video} = 1$

(c) $S = 32$: Work shown by Tsai et al indicates that this is a good value for $S$ (24).

(d) $T_1 = \frac{3*S}{4} = 24$: Work shown by Tsai et al indicates that this is a good value for $T_1$ (24).

(e) $T_2 = \frac{S}{4} = 8$: Work shown by Tsai et al indicates that this is a good value for $T_2$ (24).

(f) $L = \emptyset$.

(g) *Init*:

    i. if $index_{video} > length(I)$,
       then return,
       else continue.
    ii. $I^{mean} = mean(I(index_{video}))$: Get the mean of the current frame.
    iii. $I^{temp} = I(index_{video}) - I^{mean}$: Zero mean the current frame.
    iv. $W = \mathcal{DWT}(I_{temp}, L)$: Choose $L = 6$ to perform 6 iterations of the $\mathcal{DWT}$.
    v. $W^{temp} = W$.
    vi. $W^{temp}(1 : \frac{256}{2^L}, 1 : \frac{256}{2^L}) = 0$: Zero-out the coarse approximation.
    vii. $E$ equals the $N$ largest coefficients in $W^{temp}$: e.g. choosing $N = 6000$ allows for the embedding of 6000 audio coefficients per image frame.
    viii. $P = P$ union the current set of indices for the $N$ largest coefficients of the current frame.

(h) *Embed*:

    i. for $a = 1$ to $N$.
    ii. if $index > length(A)$,
        then GoTo *Post Processing*,
        else continue.
    iii. if $A_{index} == 0$,
        A. if $E_a \geq 0$,
           then $E_a^{mod} = E_a - |E_a| \bmod S + T_2$.
        B. else $E_a < 0$,
           then $E_a^{mod} = E_a + |E_a| \bmod S - T_2$.
    iv. else $A_{index} == 1$,
        A. if $E_a \geq 0$,
           then $E_a^{mod} = E_a - |E_a| \bmod S + T_1$. item else $E_a < 0$,
           then $E_a^{mod} = E_a + |E_a| \bmod S - T_1$.
    v. $index_{audio} = index_{audio} + 1$.

(i) *Post Processing*:

i. $W^{mod} = W^{temp} + E^{mod} + W_{mod}(1 : \frac{256}{2^L}, 1 : \frac{256}{2^L})$: Replace the coarse coefficients and replace the old values of $E$ within $W$ with $E^{mod}$.

ii. $I^{mod}(index_{video}) = \mathcal{DWT}^{-1}(W^{mod}, L)$: perform the inverse wavelet transform on $W^{mod}$.

iii. $I^{mod}(index_{video}) = I^{mod}(index_{video})$: Add the mean back into the image.

iv. $I^{mod}(index_{video}) = quantize(I^{mod}(index_{video}))$: Quantize the modified pixel values.

v. $index_{video} = index_{video} + 1$: Increment the video frame array index.

vi. GoTo *Init.*

### *F.3 1–D Extraction*

1. Description: The algorithm below defines a method to extract the audio information from those wavelet coefficients into which the audio was embedded. This algorithm is iterative.

2. Data Dictionary:

   (a) $I^{mod}$ is the set of video frames frome which the watermark information is extracted.

   (b) $W$: $W = \mathcal{DWT}(I^{mod}, L)$. Note: $\mathcal{DWT}(I^{temp}, L)$ indicates to perform $L$ iterations of the $\mathcal{DWT}$ in the data $I^{temp}$.

   (c) $W^{temp}$ is a temporary variable used to store a modified version of $W$

   (d) $E$: $E \subset W$. $E$ is the set of watermarked coefficients in $W$.

   (e) $L$ is an array that contains the indices to the locations of the largest wavelet coefficients.

   (f) $A^{recon}$ is the binary audio data extracted from $I^{mod}$.

   (g) $S$ is an integer value used for a modulus operation. The larger the value of $S$, more of an effect it has on the reconstructed image quality, but the harder it is to remove the watermark information. The smaller the value of $S$, the less impact it has on image fidelity, but the easier it is to remove the watermark information.

(h) $T_1$ is a threshold value used to embed the watermark into $I$.

(i) $T_2$ is a threshold value used to embed the watermark into $I$.

(j) $a$ is an index counter for the main *for loop*.

(k) $index_{audio}$ is an index to the current location within the array of audio coefficients.

(l) $index_{video}$ is an index to the current location within the array of video frames.

3. Functional Description:

(a) $index_{audio} = 1$

(b) $index_{video} = 1$

(c) $S = 32$: Work shown by Tsai et al indicates that this is a good value for $S$ (24).

(d) $T_1 = \frac{3*S}{4} = 24$: Work shown by Tsai et al indicates that this is a good value for $T_1$ (24).

(e) $T_2 = \frac{S}{4} = 8$: Work shown by Tsai et al indicates that this is a good value for $T_2$ (24).

(f) *Init*:

i. if $index_{video} > length(I)$,
then return,
else continue.

ii. $W = \mathcal{DWT}(I_{mod}, L)$: Choose $L = 6$ to perform six iterations of the $\mathcal{DWT}$.

iii. $W^{temp} = quantize(W)$: Quantize the wavelet coefficients.

(g) *Extract*:

i. for $a = 1$ to $X$.

ii. if $|W^{temp}(index_{audio})| \bmod S \geq \frac{T_1+T_2}{2}$,
then $A^{recon}(index_{audio} = 0$.

iii. else $|W^{temp}(index_{audio})| \bmod S < \frac{T_1+T_2}{2}$,

then $A^{recon}(index_{audio}) = 1$.

iv. $index_{audio} = index_{audio} + 1$.

(h) *Post Processing*:

i. $I^{mod}(index_{video}) = \mathcal{DWT}^{-1}(W^{temp}, L)$: Perform the inverse wavelet transform on $W^{temp}$.

ii. $I^{mod}(index_{video}) = quantize(I^{mod}(index_{video}))$: Quantize the modified pixel values.

iii. $index_{video} = index_{video} + 1$: Increment the video frame array index.

iv. GoTo *Init.*

## *Bibliography*

1. C. Sidney Burrus, Ramesh A. Gopinath, Haitao Guo. *Introduction To Wavelets and Wavelet Transforms: A Primer*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

2. Cox, I. and M.L. Miller. "A review of Watermarking and the Importance of Perceptual Modeling," *In Proceedings SPIE*, *3016*:92–99 (1997).

3. Cox, Ingemar J., et al. "Secure Spread Spectrum Watermarking for Multimedia," *IEEE Transactions on Image Processing*, *6*(12):1673–1687 (1997).

4. Daubechies, Ingrad. *Ten Lectures on Wavelets*, *61*. CBMS-SNF Regional Conference Series on Applied Mathematics. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1992.

5. Group, Joint Binary Image Experts. "Information Technology-Coded representation of pictures and audio information-Progressing bi-level image compression," *ITU-T T.82* (1993).

6. Higham, Robert A. *The 1997 Grolier Multimedia Encyclopedia*. Golier Interactive Inc., 1997.

7. Hsu, Chiou-Ting and Ja-Ling Wu. "Multiresolution Watermark for Digital Images," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, *45*(8):1097–1101 (1998).

8. Huang, Jiwa and Yun Q. Shi. "Adaptive Image Watermarking Scheme Based on Visual Masking," *Electronics Letters*, *34*(8):748–750 (1998).

9. Jayant, Nikil, et al. "Signal Compression Based on Models of Human Perception," *Proceedings of the IEEE*, *81*(10):92–99 (October 1993).

10. Jr., Roger L. Claypoole. *Adaptive wavelets transforms via lifting*. PhD dissertation, Rice University, Houston, Texas, October 1999.

11. Kaewkamnerd, N. and K.R. Rao. "Wavelet Based Image Adaptive Watermarking Scheme," *Electronics Letters*, *36*(4):312–313 (2000).

12. Lim, Jae S. *Two-Dimension Signal and Image Processing*. Prentice Hall, 1990.

13. Mallat, S. and W. Hwang. "Singularity detection and processing with wavelets," *IEEE Transactions on Information Theory*, *38*(2):617–643 (1992).

14. Mallat, S. and S. Zhong. "Characterization of signals from multiscale edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*:710–732 (July 1992).

15. Mallat, S. G. "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *11*(7):674–693 (1989).

16. Niu, Xia-Mu, et al. "Digital Watermarking of Still Images with Gray-Level Digital Watermarks," *IEEE Transactions on Consumer Electronics*, *46*(1):137–145 (2000).

17. Resnikiff, H.L. and R.O. Wells. *Wavelet Analysis*. New York: Springer-Verlag, 1998.

18. Shapiro, Jerome M. "Embedded Image Encoding Using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing*, *41*(12):3445–3462 (1993).

19. Sinha, Deepen and Ahmed H. Tewfik. "Low Bit Rate Transparent Audio Compression using Adapted Wavelets," *IEEE Transactions on Signal Processing*, *41*(12):3463–3479 (1993).

20. Sklar, Bernard. *Digital Communications: Fundamentals and Applications*. Englewood Cliffs, New Jersey 07632: PTR Prentice Hall, 1988.

21. Srinivasan, Primila and Leah H. Jamieson. "High-Quality Audio Compression Using an Adaptive Wavelet Packet Decomposition and Psychoacoustic Modeling," *IEEE Transactions on Signal Processing*, *46*(4):1085–1093 (1998).

22. Swanson, Mitchell D., et al. "Robust Audio Watermarking using Perceptual Masking," *Signal Processing*, *66*:337–355 (1998).

23. Tian, Jun and JR. Raymond O. Wells. "A Lossy Image Codec Based on Index Coding." *IEEE DCC*. 1996.

24. Tsai, Min-Jen, et al. "Joint Wavelet and Spatial Transformation for Digital Watermarking," *IEEE Transactions on Consumer Electronics*, *46*(1):241–245 (2000).

25. Vetterli, M. and C. Herley. "Wavelets and Filterbanks: Theory and Design," *IEEE Transactions on Acoustic and Speech Signal Processing*, *40*(9):2207–2232 (1992).

26. Voyatzis, G. and I. Pitas. "Applications of Toral Automorphisms on Image Watermarking," *IEEE International Conference on Image Processing*, *2*:237–240 (September 1996).

27. Wei, Z. H., et al. "Perceptual Digital Watermark of Images using Wavelet Transform," *IEEE Transactions on Consumer Electronics*, *44*(4):1267–1272 (1998).

28. Wu, Chuan-Fu and Wen-Shyong Hsieh. "Digital Watermarking Using Zerotree of DCT," *IEEE Transactions on Consumer Electronics*, *46*(1):87–94 (2000).

29. Zhang, Bo and Yuan F. Zheng. "Packed Integer Wavelet Transform Constructed by Lifting." To be published in *IEEE Transactions on Circuits and Systems for Video Technology*, 2000.

30. Zhu, Wenwu, et al. "Multiresolution Watermarking for Images and Video," *IEEE Transactions on Circuits and Systems for Video Technology*, *9*(4):545–550 (1999).

# *Vita*

Michael J. Mendenhall graduated from Oregon State University in 1996 with a B.S. in Computer Engineering. His first assignment was to Langley AFB, VA where he worked in the Air Combat Command Communications Group (ACC CG) as a Systems Analyst in the area of Ground Theater Air Control Systems (GTACS) software. Following his assignment to Langley, Captain Mendenhall was selected to attend the Air Force Institute of Technology (AFIT) at Wright-Patterson AFB, OH where he received his M.S. in Computer Engineering in March of 2001. Captain Mendenhall has a follow on assignment to Hill AFB, UT. He is assigned to the 84$^{th}$ Radar Evaluations Squadron (RADES.)

Permanent address: AFIT/ENG, 2950 P St.
Wright-Patterson AFB, OH 45433-7765

## REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 074-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
**PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

**1. REPORT DATE** *(DD-MM-YYYY)*
07-03-2001

**2. REPORT TYPE**
Master's Thesis

**3. DATES COVERED** *(From – To)*
Jan 2000 – Mar 2001

**4. TITLE AND SUBTITLE**

Wavelet-Based Audio Embedding & Audio/Video Compression

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Mendenhall, Michael, J., Captain, USAF

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)**

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

**8. PERFORMING ORGANIZATION REPORT NUMBER**

AFIT/GE/ENG/01M-18

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
AFRL/IFTA
Attn: Dr. Bob Ewing
2241 Avionics Cl
Building 620 Rm S3-A52
Wright-Patterson AFB, OH 45433
Commercial: (937) 255-6653 ext. 3592

**10. SPONSOR/MONITOR'S ACRONYM(S)**

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

With the decline in military spending, the United States relies heavily on state side support. Communications has never been more important. High-quality audio and video capabilities are a must.

Watermarking, traditionally used for copyright protection, is used in a new and exciting way. An efficient wavelet-based watermarking technique embeds audio information into a video signal. Several highly effective compression techniques are applied to compress the resulting audio/video signal in an embedded fashion.

This wavelet-based compression algorithm incorporates bit plane coding, first-difference coding, and Huffman coding. To demonstrate the potential of this audio embedding audio/video compression system, an audio signal is embedded into a video signal and the combined signal is compressed. Results show that overall compression rates of 15:1 can be achieved. The video signal is reconstructed with a median PSNR of nearly 33dB. Finally, the audio signal is extracted without error.

**15. SUBJECT TERMS**
compression, audio, video, wavelets, watermarking

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** |
|---|---|---|---|---|---|
| **a. REPORT** | **b. ABSTRACT** | **c. THIS PAGE** | | | Major Roger L. Claypoole Jr., ENG |
| U | U | U | UU | 124 | **19b. TELEPHONE NUMBER** *(Include area code)*<br>(937) 255-3636, ext 4625 |

**Standard Form 298 (Rev. 8-98)**
Prescribed by ANSI Std. Z39-18

*Form Approved*
*OMB No. 074-0188*